Simple Computer Linear Algebra System

Mikael Rask
BSc(Hons) Computer Information Systems
16th May 2005

COPYRIGHT

Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the University of Bath (see http://www.bath.ac.uk/ordinances/#intelprop).

This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed……

This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purporses of consultation.

Signed……

Acknowledgements

I would like to thank my project supervisor, Dr. Nicolai Vorobjov, for his guidance throughout the development of this project.

A special mention to my family and friends who have supported me at all times and have beard with my moaning at some critical times.

Abstract

Linear algebra is a vast area within the context of mathematics. Matrices in turn occupy a large area within the context of linear algebra. It is not complicated to deal with calculations involving matrices of small dimensions although it can become somewhat awkward. It definitely is a very tedious procedure to handle calculations with matrices of large dimensions such as 25 x 25 or even 50 x 50. Software programs have been developed to deal with these sorts of situations. However, the existing products do not relate solely to matrix related problems but to the entire linear algebra context. Thus, having to cover such a dense topic makes the programs very complex involving complex commands in some cases. In this dissertation, a simple analogy of these programs has been developed and implemented. The software program developed evolves solely around matrix-based problems. The algorithms for the main matrix operations have been implemented. These being the simple algorithms in most cases, although other more complex algorithms such as Strassen's and Gaus-Jordan have been introduced. The system has implemented a very simple and user-friendly graphical user interface and has successfully passed functionality and usability tests.

Table of Contents

1. Introduction 1
2. Literature Review 3
    2.1. Introduction 3
    2.2. Analysis of existing products 4
    2.3. The Mathematics 10
    2.4. The Algorithms 13
3. Requirements 13
  3.1. Requirements Analysis 13
    3.1.1. Existing products 15
    3.1.2 Key Questions 18
    3.1.3. Use Cases 22
  3.2. Requirements Formal Specification 22
    3.2.1. Functional Requirements 23
4. Design 27
    4.1. Programming Language 27
    4.2. System overview 28
    4.3. Input and output considerations 29
    4.4. Data-flow models 29
    4.5. Data Structures 29
    4.6. Detailed design 29
    4.7. User Interface 31
5. Implementation 33
    5.1. The Graphical User Interface 33
    5.2. The `main()` method 36
6. Testing 39
    6.1. Objective of testing 39
    6.2. Testing scope 39
    6.3. Functional/Black-box testing 40
    6.4. Usability testing 41
    6.5. Test Results Evaluation 43
7. Conclusion 45
    7.1. Future work 46
    7.2. Personal analysis 46
8. Bibliography 49
9. Appendices 51

1. Introduction

Mathematics is a world of its own. Some people are capable of dealing with mathematics in their daily life and others find it more difficult. Certain individuals do not have the whit or in some cases simply do not have the will to perform mathematical calculations by themselves. For these individuals the use of computer systems is essential to perform these calculations thrown at them in their daily life, may it be at university, in a bank or even in a top executive office.

There are a large variety of computer systems in the software market that are aimed at solving mathematical problems. Mathematics being a very extensive topic has made these programs normally to focus on particular areas within the world of mathematics. The software packages focusing on linear algebra are fairly popular and are circulating more and more everyday. However, these software packages tend to be very complex in the sense of usability as they have to cover a vast amount of material. In general, obscure commands not common to the average individual are required to be inputted in order to extract the correct functionality of the software program.

This dissertation attempts to develop a software program that narrows down the scope. The idea is to create a system that is focused exclusively to a particular area within the world of mathematics. Not only within the scope of mathematics, but within a particular area of this universal concept. The system being developed for this project evolves about the use of matrix-based linear algebra. The idea is to implement a system that solely performs mathematical calculations involving matrices.

In order to be able to develop such a system a good comprehension of the scope is required. It is very important to fully understand the mathematics that involve matrices and the different calculations that may be carried out with them. There is a substantial amount of literature available on the subject, as matrix maths is very common. Thus, before adventuring into the development of the system analysis of this literature has to be completed. In Section 2, a review is written on the literature analysed.

Having fully understood the subject, the next step is to specify what the system will need to include or in other terms the requirements of the project. Section 3 analyses and specifies the requirements for the project and software system. The requirements specification will set a base to work from. It is vital that the requirements specification is accurate and within the specified scope. Here the analysis of the literature gains relevance.

Once the requirements have been specified the next steps are the design and implementation of the system. These regularly go hand in hand as they are closely related. In Section 4, the design issues are discussed based on the requirements specified in the previous section. Specific models are used to aid the design process. Detail of how certain features will be implemented and the programming language that will be used for the implementation are also included in this section. As soon as the system has been designed the implementation process begins. Section 5 describes

the source code written to produce the software system and any issues involving the implementation process.

Having fully built the system, it is important to verify that it delivers what is expected from it. Therefore, the system has to undergo specific tests with the aim of validating the requirements specification set when commencing the project. If the requirements are met the system is valid for submission. The details about the testing procedures and their outcome are explained in Section 6.

Finally, the conclusions made about the entire project are summed up in Section 8 where an outline to future work on the system is found as well.

2. Literature Review

2.1. Introduction

The purpose of this project is to produce a simple linear algebra computer system, in essence, a simplified version of already existing systems like Maple or REDUCE. It is important to carry out thorough research on the existing products in order to produce a good and efficient system. These types of systems are of great importance to carry out complex computations that are very tedious to calculate with a pen and paper. For example, it can get very dull to calculate the inverse for a 25 x 25 matrix.

Moreover, to produce good software for this system the basic mathematics has to be understood from top to bottom, but also some more complex mathematical calculations should be comprehended. The mathematics for this system will be entirely matrix based, i.e. matrix addition, matrix multiplication, inverse matrix amongst other matrix operations.

It is also of great importance perceiving what algorithms are and how different algorithms work as well as how they affect the efficiency of programs. A good comprehension of the algorithmic concepts will also become very important when it comes to designing a good program. Detailed algorithm analysis is necessary to produce efficient software. The Strassen's algorithm is an example of the type of algorithms that have to be examined and appreciated for this project.

2.2. Analysis of existing products

Maple defines itself as "a comprehensive computer system for advanced mathematics. It includes facilities for interactive algebra, calculus, discrete mathematics, graphics, numerical computation, and many other areas of mathematics. It also provides a unique environment for rapid development of mathematical programs using its vast library of built-in functions and operations."[1]

Having used Maple to investigate its features as well as the GUI (Graphical User Interface) and functionality I have discovered that it is quite user friendly and the help menu is very useful when it comes to make any computations. Of course, Maple is a very large program and covers a vast range of mathematical areas which accounts for need of the user-friendly help menu in order for the program not to become complicated to use. The GUI is quite simple, which makes it easy for the user to feel comfortable to use the variety of functions available. The commands to carry out the computations are very straightforward and in case of any doubt, the help menu guides the user to perform the desired operation.

REDUCE is a mathematical system that is very similar to Maple in its features. The REDUCE homepage writes "it is often used as an algebraic calculator for problems that are possible to do by hand. However, the main aim of REDUCE is to support calculations that are not feasible by hand. Many such calculations take a significant

[1] Maple 8 User Manual – Introduction

time to set up and can run for minutes, hours or even days on the most powerful computers."[2]
The program being developed for this project is obviously of a much smaller dimension and only covers part of one topic covered by Maple. My program only covers some part of the linear algebra mathematics. However, I need to do something to differentiate it from existing products. Although it is still early to decide in what aspects it is going to be different, an idea could be to let the user have the ability, if desired, to select from a selection of algorithms by which the problem should be computed. Maple does not allow you to do this, moreover, the user does not know which is the algorithm used to compute the particular problem.

2.3. The Mathematics

The mathematics involved in the system being developed is entirely matrix based, and is only concerned with operations upon matrices.

DEFINITION: A matrix is a rectangular array of numbers. The numbers in the array are the elements of the matrix.[3]

Matrices are composed of rows and columns, the rows starting from the left and the columns starting at the top. So the first element in the matrix will be located in the top left-hand corner of the matrix. The elements within the matrix will be located at positions denoted as a_{ij} for matrix **A**, a_{11} being the element in the first row and the first column and a_{21} being the element in the second row and first column.

The following examples illustrate the standard notation for matrices, they are normally denoted by capital letters, say **A**, **B** or **X** for instance:

$$\mathbf{A}_{nxn} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

The program being developed will deal with matrix operations. So systems of linear equations will be expressed in matrix form and then the appropriate operations will be carried out to obtain the solutions for the system of equations, if there are any. This method of solving systems of linear equations will be explained later into more detail. First, simpler matrix operations to be carried out by the program have to be understood.

2.3.1 Matrix addition and subtraction

Matrices can be added and subtracted, so **A** + **B** = **C** or **A** – **B** = **C**. For two matrices to be added they have to be of the same dimension otherwise it is not possible so a 2 x 2 matrix cannot be added to a 2 x 3 matrix. The resulting matrix of adding matrices is obtained by the sum of each pair of corresponding elements, i.e. $c_{ij} = a_{ij} + b_{ij}$.

---

[2] http://www.uni-koeln.de/REDUCE/#intro
[3] Williams, G. (1937, p.3)

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}, \text{ so } \mathbf{A} + \mathbf{B} = \begin{bmatrix} 1+2 & 3+1 \\ 2+3 & 4+4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 5 & 8 \end{bmatrix}$$

Matrix subtraction applies the same theory as addition. Subtraction cannot occur between matrices of different dimensions and $c_{ij} = a_{ij} - b_{ij}$. Of course subtraction is not commutative, so $\mathbf{A} - \mathbf{B} \neq \mathbf{B} - \mathbf{A}$.

$$\mathbf{A} - \mathbf{B} = \begin{bmatrix} -1 & 2 \\ -1 & 0 \end{bmatrix} \qquad \mathbf{B} - \mathbf{A} = \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix}, \text{ thus } \mathbf{A} - \mathbf{B} \neq \mathbf{B} - \mathbf{A}.$$

2.3.2. Matrix scaling and multiplication

Matrices can be scaled by multiplying the matrix by a fixed factor, so each and every element of the matrix is multiplied by this factor. The resulting scaled matrix will be of the same dimension as the original matrix. The notation for scaling matrices is x**A** = **X**, **A** being the original matrix, x being the scalar factor and **X** being the resulting scaled matrix. This scaling technique helps us to negate matrices e.g. (-1)**A** = -**A**. Using this approach we can now look at subtraction from another viewpoint, by adding one matrix to the negation of the other i.e. **A** – **B** = **A** + (-1)**B**.

$$3\mathbf{A} = 3\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ 6 & 12 \end{bmatrix}, \text{ the scalar x = 3.}$$

Matrix multiplication is somewhat more complicated than addition, subtraction and scaling because it does not involve straightforward operations between corresponding elements in matrices. Also, it is not limited to matrices of the same dimensions. Moreover, for multiplication to possible to be computed the number of rows in one matrix has to be equal to the number of columns in the other. This means that if two matrices are of identical dimensions, unless they are square matrices i.e. same number of rows and columns, multiplication cannot occur. This is because for two 2 x 3 matrices the number of rows does not equal the number of columns, however if one matrix was a 2 x 3 and the other a 3 x 2 the multiplication between these two matrices can be calculated. In order to calculate the multiplication correctly, the elements in the row from the first matrix say **A** in the form $[a_{i1}\ a_{i2}\ a_{i3}\ a_{in}]$ have to be multiplied with the corresponding elements in the column of the second matrix say **B** in the form $\begin{bmatrix} b_{ij} \\ b_{2j} \\ b_{3j} \\ b_{nj} \end{bmatrix}$ and added together to give the element c_{ij} in the resulting matrix **C**.

So $c_{ij} = a_{i1}b_{ij} + a_{i2}b_{2j} + a_{i3}b_{3j} + a_{in}b_{nj}$. The order in which the matrices are multiplied together is therefore very important; matrix multiplication is not always commutative, thus, normally **AB** ≠ **BA**.

$$\mathbf{AB} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}, \text{ so a11= 1*2 + 3*3, a12 = 1*1 + 3*4 etc } \therefore \mathbf{AB} = \begin{bmatrix} 11 & 13 \\ 16 & 18 \end{bmatrix}.$$

2.3.3. Identity Matrix

DEFINITION: Identity Matrix – an identity matrix is a square matrix with 1s in the diagonal locations (1,1), (2,2), (3,3) and so on and zeros elsewhere. We write $\boldsymbol{I_n}$ *for the n x n identity matrix. The following are examples of identity matrices.*[4]

$$\mathbf{I_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{I_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.3.4. Gauss-Jordan Elimination for solving systems of linear equations

As mentioned previously, the solutions for a system of linear equations can be deduced using matrices. For the following example of a system of linear equations the solutions for x1, x2 and x3 can be worked out by associating the system with a matrix.

$$\begin{aligned} x_1 - x_2 + 3x_3 &= 3 \\ 2x_1 - x_2 + 2x_3 &= 2 \\ 3x_1 + x_2 - 2x_3 &= 2 \end{aligned}$$

The initial matrix derived from the system of equations is called the augmented matrix and is composed of a combined matrix with the matrix coefficients of the variables (say **A**) and the column matrix of constant terms (say **B)**, i.e. [A:B]. So for the above system of equations the augmented matrix is $\begin{bmatrix} 1 & -1 & 3 & 3 \\ 2 & -1 & 2 & 2 \\ 3 & 1 & -2 & 2 \end{bmatrix}$. Once we have the augmented matrix a series of row operations will be carried out on the matrix in order to progressively eliminate variables. The row operations are formally called elementary row operations and are of the nature of: adding/subtracting rows (and multiples of rows) together, exchanging positions of rows etc. The objective is to carry out the row operations until the matrix is in reduced echelon form.

DEFINITION: A matrix is in reduced echelon form if

1. *Any rows consisting entirely of zeros are grouped at the bottom of the matrix.*
2. *The first non-zero element of each other row is 1. This element is called a* ***leading 1****.*
3. *The leading 1 of each row after the first is positioned to the right of the leading 1 of the previous row.*
4. *All other elements in a column that contains a leading 1 are zero.*[5]

---

[4] Williams, G. (1937, p.6)
[5] Williams, G. (1937, p.16)

So a matrix in reduced echelon form is basically composed of an identity matrix combined with a column matrix of the solutions to the system of linear equations, i.e. $[I_n:X]$, for example $\begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & x_2 \\ 0 & 0 & 1 & x_3 \end{bmatrix}$.

In essence, the system of linear equations is being converted to an equivalent system with the same solutions. This method to solve a system of linear equations is known as Gauss-Jordan Elimination. The fully worked out procedure using Gauss-Jordan elimination for this system is as follows.

$$\begin{bmatrix} 1 & -1 & 3 & 3 \\ 2 & -1 & 2 & 2 \\ 3 & 1 & -2 & 2 \end{bmatrix} \underset{\substack{R_2 - 2R_1 \\ R_3 - 3R_1}}{\rightarrow} \begin{bmatrix} 1 & -1 & 3 & 3 \\ 0 & 1 & -4 & -4 \\ 0 & 4 & -11 & -7 \end{bmatrix} \underset{\substack{R_1 + R_2 \\ R_3 - 4R_2}}{\rightarrow} \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & -4 & -4 \\ 0 & 0 & 5 & 9 \end{bmatrix}$$

$$\underset{\rightarrow}{R_3/5} \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & -4 & -4 \\ 0 & 0 & 1 & 9/5 \end{bmatrix} \underset{\substack{R_1 + R_3 \\ R_2 + 4R_3}}{\rightarrow} \begin{bmatrix} 1 & 0 & 0 & 4/5 \\ 0 & 1 & 0 & 16/5 \\ 0 & 0 & 1 & 9/5 \end{bmatrix}$$

So the equivalent system of equations that we have obtained is as follows:

$$\begin{aligned} x1 &= 4/5 \\ x2 &= 16/5 \\ x3 &= 9/5 \end{aligned}$$

Once familiar with the Gauss-Jordan elimination method, we can introduce the inverse matrix.

2.3.5. Inverse Matrix

*DEFINITION: Let **A** be an n x n matrix. If a matrix **B** can be found such that **AB** = **BA** = $\boldsymbol{I_n}$, then **A** is said to be invertible and **B** is called the inverse of **A**. If such a matrix **B** does not exist, then **A** has no inverse.*[6]

The Gauss-Jordan elimination method can be used to find the inverse of matrices if they have one. To find the inverse of a matrix, first a combined matrix composed of the original matrix attached with the identity matrix, $\mathbf{I}_n$, has to be formed, i.e. $[\mathbf{A}:\mathbf{I}_n]$. Then the Gauss-Jordan elimination technique can be carried out to obtain the reduced echelon form of the matrix. This is illustrated with the following example:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 4 & 5 & 3 \end{bmatrix} \qquad \mathbf{A}^{-1} = ?$$

[6] Williams, G. (1937, p.78)

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 4 & 5 & 3 & 0 & 0 & 1 \end{bmatrix} \quad R_3 - 4R_1 \rightarrow \quad \begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & -3 & -9 & -4 & 0 & 1 \end{bmatrix}$$

$$R_1 - 2R_2 \rightarrow \quad \begin{bmatrix} 1 & 0 & -1 & 1 & -2 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & -3 & -9 & -4 & 0 & 1 \end{bmatrix} \quad R_3 + 3R_2 \rightarrow \quad \begin{bmatrix} 1 & 0 & -1 & 1 & -2 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & -3 & -4 & 3 & 1 \end{bmatrix}$$

$$-R_3/3 \rightarrow \quad \begin{bmatrix} 1 & 0 & -1 & 1 & -2 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 4/3 & -1 & -1/3 \end{bmatrix} \quad \begin{matrix} R_1 + R_3 \\ \rightarrow \\ R_2 - 2R_3 \end{matrix} \quad \begin{bmatrix} 1 & 0 & 0 & 7/3 & -3 & -1/3 \\ 0 & 1 & 0 & -8/3 & 3 & 2/3 \\ 0 & 0 & 1 & 4/3 & -1 & -1/3 \end{bmatrix}$$

Therefore, $\mathbf{A}^{-1} = \begin{bmatrix} 7/3 & -3 & -1/3 \\ -8/3 & 3 & 2/3 \\ 4/3 & -1 & -1/3 \end{bmatrix}$.

For large matrices, this process can take a long time and in the end it might be possible that the matrix is not invertible, i.e. does not have matrix. So before computing the inverse, the determinant of the matrix can be calculated. If it happens that the determinant is zero then the matrix does not have an inverse.

2.3.6 The determinant

The notation for the determinant of a matrix is somewhat different, it is represented with vertical lines, $|\mathbf{A}|$ for example, and the result is a number not a matrix. The determinant can be a concept that is difficult to grasp because there are various ways of calculating the determinant and some methods are only applicable for matrices of certain dimensions. It is simple to find the determinant for 2 x 2 matrices, and slightly more complex for 3 x 3 matrices.

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{matrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{32}a_{23}$$

However, neither of these approaches can be used to calculate the determinant for matrices n x n, where n > 3. The cofactor method can be used for n x n matrices, when ≥ 3.

DEFINITION: The determinant of a square matrix is the sum of the products of the elements of the first row and their cofactors.[7]

[7] Williams, G. (1937, p.121)

*If **A** is 3 x 3,* $|\boldsymbol{A}| = a11\,C11 + a12C12 + a13C13$

*If **A** is 4 x 4,* $|\boldsymbol{A}| = a11\,C11 + a12C12 + a13C13 + a14C14$

$\vdots$

*If **A** is n x n,* $|\boldsymbol{A}| = a11\,C11 + a12C12 + a13C13 + \ldots + a1nC1n$

In order to understand the notation used above, the terms minor and cofactor are to be understood.

*DEFINITION: Let **A** be a square matrix,*[8]

*the minor of the element aij is denoted Mij and is the determinant of the matrix that remains after deleting row i and column j of **A**, and*
the cofactor of aij is denoted Cij and is given by $Cij = (-1)^{i+j}Mij$

An illustration of Mij and Cij for $\mathbf{A} = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 2 & 2 \\ 3 & 1 & 3 \end{bmatrix}$,

$$M11 = \begin{vmatrix} 1 & 3 & 1 \\ 2 & 2 & 2 \\ 3 & 1 & 3 \end{vmatrix} = \begin{vmatrix} 2 & 2 \\ 1 & 3 \end{vmatrix} = (2*3) - (2*1) = 4 \qquad C11 = (-1)^{1+1}M11 = (-1)^{1+1}4 = 4$$

$$M21 = \begin{vmatrix} 1 & 3 & 1 \\ 2 & 2 & 2 \\ 3 & 1 & 3 \end{vmatrix} = \begin{vmatrix} 3 & 1 \\ 1 & 3 \end{vmatrix} = (3*3) - (1*1) = 8 \qquad C21 = (-1)^{2+1}M21 = (-1)^{2+1}8 = -8$$

The determinant is define in the definition for the first row, can be calculated expanding along any row or column. It makes it more convenient to expand along the row or column with most 0's.

In the following section algorithms and their efficiency is discussed. So for reasons explained later on, the algorithm for calculating the determinant in the way explained above is not efficient at all. So for computational purposes, there is an alternative method to calculate the determinant of a matrix. This method is known as the upper triangular form and uses Gauss-Jordan elimination.

DEFINITION: A square matrix is called an upper triangular matrix is all the elements below the main diagonal are zero.[9]

The Gauss-Jordan elimination is used to convert the matrix, for which the determinant is going to be computed for, to the upper triangular form. Once the matrix is in this form the determinant can be evaluated straight away by multiplying together the elements in the diagonal.

---

[8] Williams, G. (1937, p.120)
[9] Williams, G. (1937, p.136)

2.4. The Algorithms

Having thoroughly understood the mathematics underlying this project and decided what mathematical operations are going to be carried out by the program, it is important to decide what algorithms are going to be used to compute these operations. In order to produce an efficient program the algorithms should be carefully designed.

DEFINITION: An algorithm is a procedure consisting of a finite set of unambiguous rules which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems.[10]

We are surrounded by algorithms in our every day life, if we think about it carefully; many things that we carry out daily involve an algorithm. For example, something as simple as making a cake involves an algorithm. In this particular case the input are the ingredients, the output is the cake and the recipe is the algorithm that we follow to convert the ingredients into the cake. This example helps to understand the nature of algorithms from a different perspective and not only in computer jargon.

So more informally, algorithms are a series of explicit instructions that take an input in the form of some value and produce an output. Basically, an algorithm converts an input of values into an output.[11] However, algorithms do not necessarily produce the correct output; in fact, algorithms may not generate an output at all. These algorithms are not very useful. Even if an algorithm does produce an output, this algorithm may not particularly be very efficient for the purpose of the program. Thus, when constructing a program it is important to carry out a thorough algorithm analysis to decide which algorithms are most appropriate for the creation of the program. The algorithm analysis is an important part of the program design phase.

2.4.1. Efficiency and Complexity

When constructing programs, it is desirable that the program is as efficient as possible. In computer jargon when we talk about program or algorithm efficiency we mean how fast the program or algorithm is, as the problem increases in size. However, we do not calculate or relate efficiency to normal time measurements, like seconds, minutes etc. This is because the time needed to run the algorithm depends on various external factors such as the programming language, the compiler, and the processor of the computer amongst others. The efficiency has more to do with the number of operations that are computed for a determined problem and how these operations increase as the problem increments in size. So efficiency is described as a function of the problem size.

The complexity of an algorithm is determined by the number of operations that the algorithm has to go through to perform the whole computation. So when adding two matrices the number of addition operations performed to obtain the result matrix determines its complexity. So for two n x n matrices being added to give a third n x n matrix the number of operations to perform the addition is n^2 because we are adding

[10] Kronsjö L. (1987, p.1)
[11] Cormen T. H. (2001, p.5)

corresponding elements to produce an n x n matrix. The number of operations for the subtraction algorithm is the same.
It is logical to believe that the number of computations for matrix multiplication is larger than for addition or subtraction. This is because when computing the matrix multiplication first we have multiplication operations which are then added together, which introduces addition operations. The whole matrix multiplication process has been explained earlier in section 3. The complexity of the matrix multiplication algorithm works out to be n^3. The complexities of algorithms like the ones explained here do not have much significance standing on their own. Complexity and efficiency of algorithms are very closely related. In general, these are inversely proportional. So as the complexity of the algorithms increases its efficiency decreases. Therefore complexities of algorithms become significant when there is more than one method or algorithm to solve a particular problem and their complexities can be compared. The comparison of the complexity of two or more algorithms with the same final outcome will allow for the implementation of the most efficient one.
For example, Strassen's algorithm is also an algorithm to perform matrix multiplication. If compared with the previously described algorithm for matrix multiplication it is found that it is slightly more efficient. Strassen's algorithm has an efficiency function of $n^{\log 7}$ which works out to be less than n^3. So in theory, Strassen's algorithm is more efficient, but only in theory because in practice in some cases the story changes. "Strassen's algorithm is often not the best method of choice for matrix multiplication, for four reasons: the constant factor hidden in the running time of Strassen's algorithm is larger than the constant factor in the naïve $\Theta(n^3)$ method, when the matrices are sparse methods tailored for sparse matrices are faster, Strassen's algorithm is not quite as numerically stable as the naïve method and the sub-matrices formed on the levels of recursion consume space."[12] The 'naïve' method refers to the basic matrix multiplication method explained in section 3. Strassen's algorithm is usually used for very large matrices where as the 'naïve' method is used for smaller ones.

DEFINITION: Strassen's Algorithm [3]

$$\textit{If } \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} m_1 + m_2 - m_4 + m_6 & m_4 + m_5 \\ m_6 + m_7 & m_2 - m_3 + m_5 - m_7 \end{bmatrix}, \textit{ where}$$

$m_1 = (a_{12} - a_{22})(b_{21} + b_{22})$ $\quad m_4 = (a_{11} + a_{12})b_{22}$ $\quad m_7 = (a_{21} + a_{22})b_{11}$
$m_2 = (a_{11} - a_{22})(b_{11} + b_{22})$ $\quad m_5 = a_{11}(b_{12} - b_{22})$
$m_3 = (a_{11} - a_{21})(b_{11} + b_{12})$ $\quad m_6 = a_{22}(b_{21} - b_{11})$

Strassen's algorithm reduces by 1 the number of multiplications with respect to the naïve method but increments the additions by 14. The reduction of 1 multiplication compensates for the extra addition for very large matrices.

As explained in section 3, there is an alternative method to the standard definition to evaluate the determinant in order to make the computation feasible within reasonable time for large matrices. The efficiency function for the standard form is n! which is highly inefficient for a large n.

[12] Cormen T. H. (2001, p.740)

Table 2.1 – A comparison of the growth of some common complexity functions.[13]

Time complexity function	Problem size n			
	10	10^2	10^3	10^4
$\log_2 n$	3.3	6.6	10	13.3
N	10	10^2	10^3	10^4
$n \log_2 n$	0.33×10^2	0.7×10^3	10^4	1.3×10^5
n^2	10^2	10^4	10^6	10^8
n^3	10^3	10^6	10^9	10^{12}
2^n	1024	1.3×10^{30}	$> 10^{100}$	$> 10^{100}$
n!	3×10^6	$> 10^{100}$	$> 10^{100}$	$>10^{100}$

As this table shows an algorithm with complexity n! is by far amongst the least effective algorithms and in practice is almost unfeasible for an n of size 10^4.

[13] Kronsjö L. (1987, p.4)

3. Requirements

3.1. Requirements Analysis

In order to construct and implement a system with the correct features required by the user a thorough analysis of the requirements has to be conducted. A complete comprehension of the problem description is important in order to meet the user's needs. It must be clear what features need to be included and excluded, as well as other factors such as the expertise of the user amongst others.

Potential users of the system contributed towards the compilation of the requirements analysis. They were requested to answer specific target questionnaires which helped to understand the users' point of view and key aspects critical to the system. (Refer to the appendix) Also, the analysis of existing products of similar nature contributes to the comprehension of features to be included, disregarded or even developed further.

3.1.1. Existing products

The analysis of the existing products carries on from the analysis performed during the literature review. As discussed earlier, there are many products in the software market that perform algebraic computations. More precisely programs that solve matrix-based linear algebra problems which are the ones we are interested in for the development of this system. Two current systems, Axiom and Maple 8, have been looked at closely and their key features have been analysed. These two products have been chosen to be analysed principally because they differ considerably from one and other in terms of their user interfaces.

Axiom is a console program, where as Maple has developed a graphical user interface (GUI). This has been illustrated below:

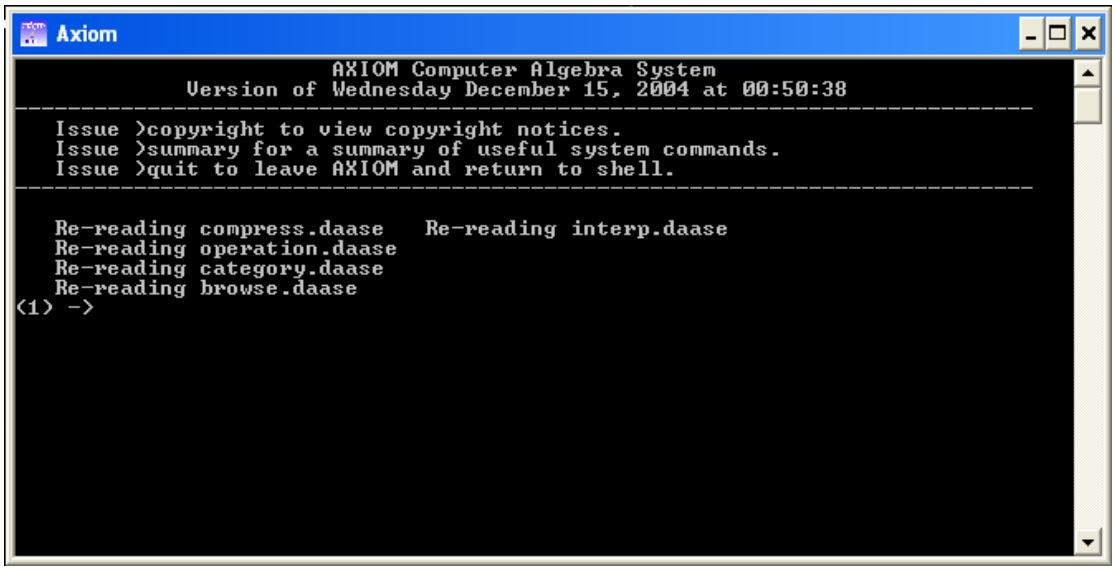

Axiom
AXIOM Computer Algebra System
Version of Wednesday December 15, 2004 at 00:50:38
Issue)copyright to view copyright notices.
Issue)summary for a summary of useful system commands.
Issue)quit to leave AXIOM and return to shell.
Re-reading compress.daase
Re-reading interp.daase
Re-reading operation.daase
Re-reading category.daase
Re-reading browse.daase
(1) ->

Figure3.1. Screen shot of Axiom when initialised.

maple 8
command the brilliance of a thousand mathematicians
Waterloo Maple
ADVANCING MATHEMATICS
© 2002 Waterloo Maple Inc. Maple is a registered trademark of Waterloo Maple Inc.

Figure3.2. Screen shot of Maple 8 when initialised.

Maple 8 is much more user friendly than Axiom. Axiom, being a console program is very complicated to use in terms of interaction with the program, especially because nowadays the majority of the software products in the market use sophisticated GUIs. The average computer user struggles when it comes to use a console program where everything is typed in and there is no aid of devices such as the mouse. It is not clear, once Axiom has been initialised, of how to carry out any mathematical operations and no directions to a help menu or user guide are available. One must go online in order to figure out how to perform any computations. The commands are reasonably simple and once one is familiar with them it quite straight forward to solve any mathematical problems. Even though Axiom is a console program, the results are displayed in fairly readable format.

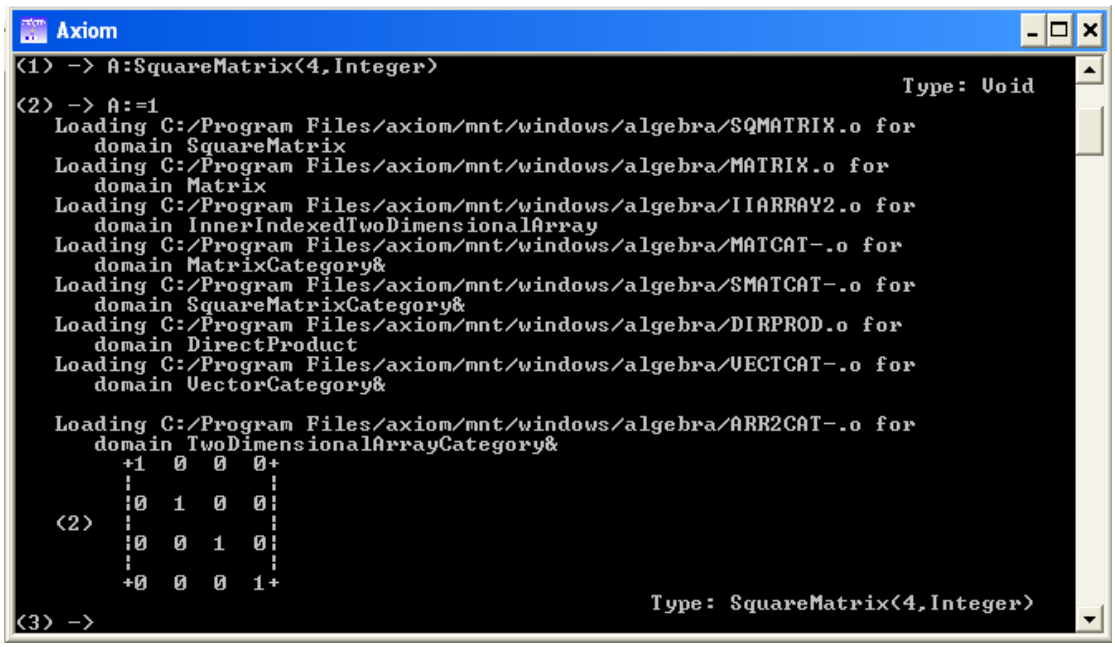


Figure3.3. Screen shot from Axiom depicting the display of a matrix.

Maple 8 is a totally interactive program. The GUI makes Maple very user-friendly and the user finds it easier to work with in comparison to Axiom. Maple offers the user an interactive help guide which can be consulted simultaneously while working on a computation. Once again, when familiar with the commands it is relatively simple to work with Maple to solve any mathematical problems. The solutions are displayed in an extremely appropriate and clear manner.

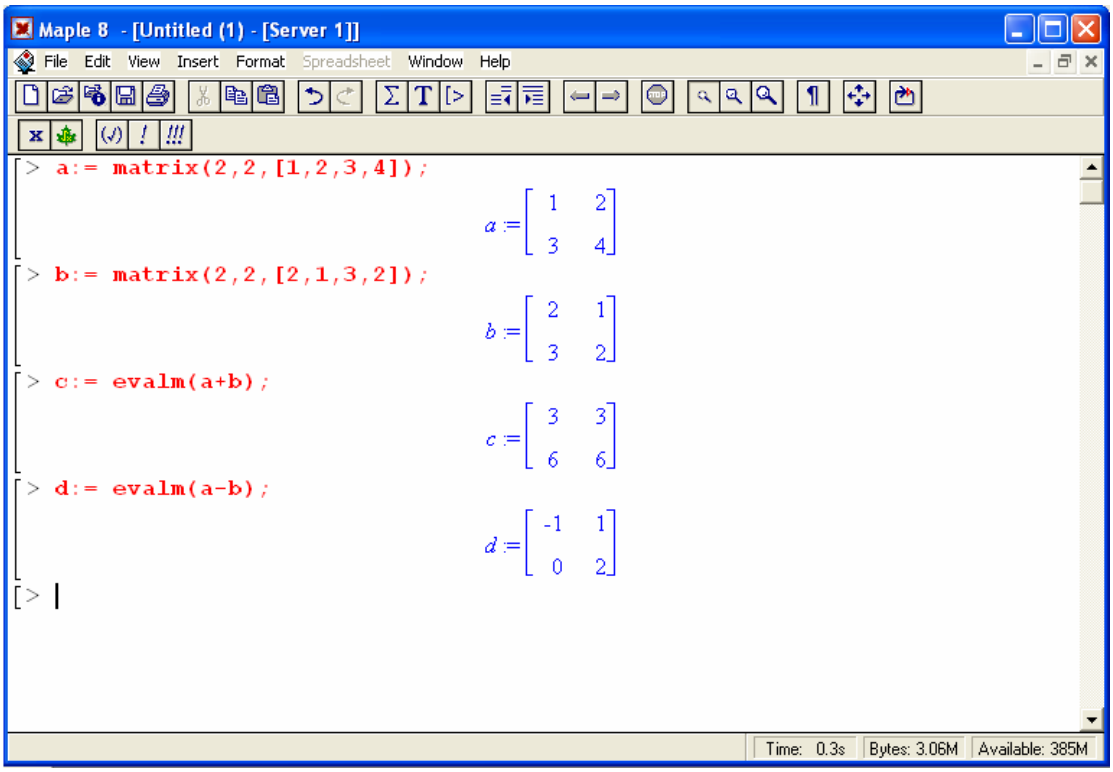

Maple 8 - [Untitled (1) - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
> a:= matrix(2,2,[1,2,3,4]);
> b:= matrix(2,2,[2,1,3,2]);
> c:= evalm(a+b);
> d:= evalm(a-b);
Time: 0.3s Bytes: 3.06M Available: 385M

Figure3.4. Screen shot depicting commands and display of computations in Maple.

3.1.2 Key Questions

Having analysed existing products in the market of similar characteristics to the system being developed and having interviewed potential users of the future system it is time for the developer of the system to answer some key questions about the system in order to compile the final requirement specification document that will be followed strictly when it comes to the implementation stage. These are as follow:

- What is the objective of the system?
- Are the any software and hardware limitations?
- What type of user is the system targeted at?
- What is the input data of the system?
- What sort of UI is required for the system?
- How will the system handle any error?
- How is the system going to be tested?
- Does the system need to consider any security issues?
- Will the users require any training once the system is complete and ready to be run?

What is the objective of the system?

The linear algebra system is intended to work as calculator. It will emulate existing systems such as Axiom, Reduce or Maple but only to a small-scale. The computations carried out by the system will be entirely matrix based, i.e. matrix arithmetic. This includes:

- Matrix addition
- Matrix subtraction
- Matrix multiplication
- Matrix inverse
- Matrix determinant
- Solving systems linear of equations

Simple algorithms will be implemented in the system to carry out these arithmetic operations. However, there will be autonomy to implement more complex algorithms to reduce complexity for certain computations depending on time constraints, for example, implementing Strassen's algorithm to solve matrix multiplication. A further option is to provide the user with the opportunity to select the algorithm he/she desires the computation to be carried out by.

Are the any software and hardware limitations?

The programming language that has been recommended to implement the system is Java as an alternative to other programming languages such as C/C++. The programmer is more familiar with Java thus making it simpler to be implemented. Also Java is an object oriented language which will allow for more dynamic programming. Java offers two important factors that may be necessary if the system was to be developed further into a more complex and professional oriented system. These are safety and portability.

There isn't a constraint on memory so the efficiency of the algorithms is not a key factor in the system. However, the system should perform within normal time restrictions, so the algorithms implemented should be as efficient as possible. Java uses a memory management scheme which allows for the best use of the memory available.

The hardware to be used is any available PC in the library that supports a Java compiler.

What type of user is the system targeted at?

The system is only of small-scale and will be targeted at A-level or University undergraduate students, not for professional use. The target of the system is mainly to allow the user to verify the results of certain matrix calculations as sometimes large matrices are involved and it may be difficult to assert that the final result of a calculation is correct. The system is therefore useful for producing results of calculations, it will not explain how the result to the calculation has been obtained, i.e. it is not a teaching system.

What is the input data of the system?

The data inputted into the system is basically mathematical notation in the form of matrices or systems of linear equations. The user will have to input the values of the matrices manually. Nonetheless, it is optional for the system to read from a file. If the matrix is very large it would become very tedious to input it manually, say a 50 x 50 matrix. It will be important to specify the maximum size of the matrices in order not to increase the computation time drastically and also the size of the numbers in the matrices to avoid overflow. The software will then handle the information inputted and will output the solution if it exists in an appropriate format.

What sort of UI is required for the system?

The user interface has been decided to be very simple. This is due to the restrictions of the programming skills of the programmer. For this system it is not necessary to implement a highly sophisticated interface as the importance lies on the mathematical side and retrieving the correct answers to mathematical problems. A very simple graphical user interface (GUI) will be implemented in detriment to a console program. This is because in the current times the average user will find it complicated to interact with a console program. A simple GUI is much more user-friendly, which is a key issue nowadays when building an interactive system.
An initial system menu should be made available in order to make it simple for the user to carry out the desired mathematical operations. The instructions should be straight forward to follow in order to avoid any confusion in the user.

How will the system handle any errors?

The system should be able to deal with error handling. If any erroneous commands are inputted into the system it should recognise them and prompt with an error message and allow the user to amend the error. The errors may be such as introducing incorrect size of matrices or inputting the wrong type of character, i.e. inputting a letter instead of number.
Overflow should be taken into account and the user should be informed with the largest number that can be inputted into the matrices.

How is the system going to be tested?

The testing procedure will be recursive and will take place throughout the implementation process, although it will be expanded once the system has been completed. However, the testing process will not be extremely intense due to time constraints and also because the nature of the system at this point in time, i.e. not being a safety-critical application will not require the testing process to include all types of testing.

The testing process will essentially consist of defect testing. Sommerville explains that “the goal of defect testing is to expose latent defects in a software system before the system is delivered”[14] This involves testing for the functionality of the system. The aim is to test against the requirement specification and ensure that the essential

[14] Sommerville – Software engineering

requirements have been met. The principal type of testing that will be carried out is black-box testing and if the project is not pushed for time some white-box testing will be completed too. A test plan, with a series of test cases, should be drawn up in order to facilitate the testing procedure.

Does the system need to consider any security issues?

The system being developed for this project will not require any security requirements as no confidential information will be handled with. No sort of access restrictions will be implemented into the system.

Will the users require any training once the system is complete and ready to be run?

The user will have a mathematical background so the only training that may be required is to become familiar with how to use the system. The user guide will help the users learn how to perform the desired operations.

3.1.3. Use Cases

The use cases act as a walkthrough predetermining how the system will react to user actions and are a means of aid to compile the final requirements specification.

Use Case 1

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Main Success Scenario:
1. The student opens the LAS.

Extensions:
1. a. Student fails to execute LAS.

Use Case 2

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 1
Main Success Scenario:
1. The student enters the number of matrices desired.
2. The student enters size of matrices.
3. The student enters matrix data.
4. Matrices are displayed.

Extensions:
1a. Student enters unrecognisable data.
    1a1. System prompts with an error message and directs user to enter data again.
2a. Student enters unrecognisable data.
    2a1. System prompts with an error message and directs user to enter data again.
3a. Student enters unrecognisable data.

3a1. System prompts with an error message and directs user to enter data again.

4a. LAS does not display matrix.

4a1. Error message appears. User asked to resume computation.

Use Case 3

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 1
Main Success Scenario:

1. Student enters the number of matrices desired.
2. Student opens and sends file with matrices to the LAS.
3. LAS displays matrices.

Extensions:

1a. Student enters unrecognisable data.

1a1. System prompts with an error message and directs user to enter data again.

2a. Student's file fails to be read by LAS.

2a1. File may be corrupted, LAS prompts to import file again.

3a. LAS does not display matrix.

3a1. Error message appears. User asked to resume computation.

Use Case 4

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student decides to modify a matrix.
2. Student selects modify option from menu.
3. Student enters new matrix.
4. LAS displays new matrix.

Extensions:

2a. Student fails to select modify option.

3a. Student enters unrecognisable data.

3a1. System prompts with an error message and directs user to enter data again.

4a. LAS does not display matrix.

4a1. Error message appears. User asked to resume computation.

Use Case 5

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to add the matrices from the option menu.
2. LAS adds matrices.

3. LAS displays result.

Extensions:

2a. LAS fails to add matrices; Matrices are of different sizes and addition cannot be performed
    2a1. LAS informs user and requests to start again.
3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 6

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to subtract one matrix from the other from the option menu.
2. LAS subtracts matrices.
3. LAS displays result.

Extensions:

2a. LAS fails to perform subtraction; Matrices are of different sizes and subtraction cannot be performed.
    2a1. LAS informs user and requests to start again.
3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 7

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Students chooses to multiply matrices together using naïve algorithm from the option menu.
2. LAS multiplies matrices together using naïve algorithm.
3. LAS displays result.

Extensions:

2a. LAS fails to perform multiplication; Matrices are of incorrect sizes and multiplication cannot be performed.
    2a1. LAS informs user and asks user to start again.
3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 8

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to multiply matrices together using Strassen's algorithm from the option menu.
2. LAS multiplies matrices together using Strassen's algorithm.

3. LAS displays result.

Extensions:

2a. LAS fails to perform multiplication; Matrices are of incorrect sizes and multiplication cannot be performed.
    2a1. LAS informs user and asks user to start again.

3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 9

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to find the inverse of a matrix from the option menu.
2. LAS finds inverse of matrix.
3. LAS displays result.

Extensions:

2a. LAS fails to find inverse; Matrix does not have inverse.
    2a1. LAS informs user and asks user to start again.

3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 10

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to find the determinant of a matrix from the option menu.
2. LAS finds determinant of matrix.
3. LAS displays result.

Extensions:

2a. LAS fails to find determinant.
    2a1. LAS informs user and asks user to start again.

3a. LAS does not display matrix.
    3a1. Error message appears. User asked to resume computation.

Use Case 11

Primary Actor: Maths student
Scope: Linear Algebra System (LAS)
Precondition: Use Case 2 or 3
Main Success Scenario:

1. Student chooses to solve a system of linear equations from the option menu.
2. LAS solve system of linear equations.
3. LAS displays result.

Extensions:

2a. LAS fails to solve system of linear equations; System does not have solutions.
    2a1. LAS informs user and asks user to start again.

3a. LAS does not display result.
3a1. Error message appears. User asked to resume computation.

3.2. Requirements Formal Specification

In order to make the requirements specification clear it is important to clarify some of the jargon used to specify the requirements that are absolutely essential and those that are of a lesser importance or say optional. Some specific requirements will be crucial for the project to go ahead; others just complement the project to make it more complex.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, at http://www.armware.dk/RFC/rfc/rfc2119.html.

3.2.1. Functional Requirements

3.2.1.1 System Requirements

3.2.1.1.1. The system SHALL carry out matrix based computations.

3.2.1.1.2. The system MUST be able to compute matrix additions, subtractions and multiplications as well as finding matrix inverses and determinants and solving systems of linear equations, where the matrices allows for this.

3.2.1.1.3. It is OPTIONAL to implement any other matrix arithmetic operations.

3.2.1.1.4. It is OPTIONAL for the system to offer a number of different paths, through which the computation can be carried out, to the user to choose from.

3.2.1.1.5. The system SHALL NOT explain how the result to the calculation has been obtained, as it's not a teaching centre.

3.2.1.1.6. The system MAY include an option menu in order for the user to choose which operation to perform.

3.2.1.2. Implementation requirements

3.2.1.2.1. The system SHALL implement the simple and naïve algorithms for these arithmetic operations.

3.2.1.2.2. The system MAY implement more complex algorithms to increase efficiency.

3.2.1.3. Input requirements

3.2.1.3.1. The input data MUST be in the form of matrices.

3.2.1.3.2. The user SHALL input the data manually into the system.

3.2.1.3.3. It is OPTIONAL to allow the user to input data from a file.

3.2.1.3.4. The user SHOULD be able to input at least 50 x 50 sized matrices.

3.2.1.4. User Interface Requirements

3.2.1.4.1. The user interface is SHOULD be very simple.

3.2.1.4.2. The user interface is RECOMMENDED to be a simple graphical user interface.

3.2.1.4.3. The resulting output MUST be displayed in an understandable and appropriate manner.

3.2.1.5. Error handling requirements

3.2.1.5.1. The system SHOULD deal with input errors. If the user inputs erroneous data the system SHOULD warn the user with an error message advising how to follow.

3.2.1.5.2. Any error messages MUST be informative, written in a polite manner and non-intrusive.

3.2.2. Non-functional Requirements

3.2.2.1. Software and hardware requirements

3.2.2.1.1. The system is RECOMMENDED to be implemented in Java.

3.2.2.1.2. The algorithms SHOULD be as efficient as possible.

3.2.2.1.3. The system SHOULD perform with normal time restrictions.

3.2.2.1.4. The system MUST run on any free-standing PC in the library that's supports a Java compiler.

3.2.2.2. User requirements

3.2.2.2.1. The system SHOULD support a computer literate user.

3.2.2.2.2. The system SHALL be targeted at A-level and university undergraduate students.

3.2.2.2.3. The system SHALL NOT be implemented for professional use.

3.2.2.3. User Interface requirements

3.2.2.3.1. The interaction with the system SHOULD be straightforward.

3.2.2.4. Testing requirements

3.2.2.4.1. The testing approach taken SHOULD be defect testing; black-box testing MUST be performed and it is RECOMMENDED to perform white-box testing too.

3.2.2.4.2. The system SHALL undergo usability tests.

3.2.2.4.3. A test plan MUST be outlined to test for the minimum requirements to give validity to the system.

3.2.2.5. Security requirements

3.2.2.5.1. The system SHALL not consider any security requirements.

3.2.2.5.2. The system SHOULD implement any access restrictions.

3.2.2.6. Training requirements

3.2.2.6.1. No training SHOULD be necessary for the use of the system.

3.2.2.6.2. A user guide MUST be made available on paper and MAY be integrated into the system.

3.2.2.7. Documentation requirements

3.2.2.7.1. The documentation SHOULD follow the structure specified by the Project Supervisor. The outline being:

- Title page
- Abstract
- Introduction
- Requirements
- Design
- Implementation
- Testing
- Critical Evaluation
- Appendix

3.2.2.7.2. The final project document MUST be thermally bound.

3.2.2.8. Delivery requirements

3.2.2.8.1. The project documentation together with the software program developed MUST be submitted by 4 pm. on Monday 16th May in the Computer Science Department Office.

4. Design

The design process begins immediately after the requirements specification has been compiled and agreed to by developer and customer. In this section we discuss the design issues taken into consideration in order to develop the linear algebra system according to the specified requirements. A fairly high-level overview of the system is explained, however, certain areas are described into more detail. First of all the selection of the programming language used to implement the system is justified. The design may vary due to the limitations of a programming language.

At the first stage of the design process it is important to identify the objects or instances that are going to make up the system. These can be extracted from the requirements analysis. Identifying these instances will facilitate the whole design process as they will set a base from which to build up on.

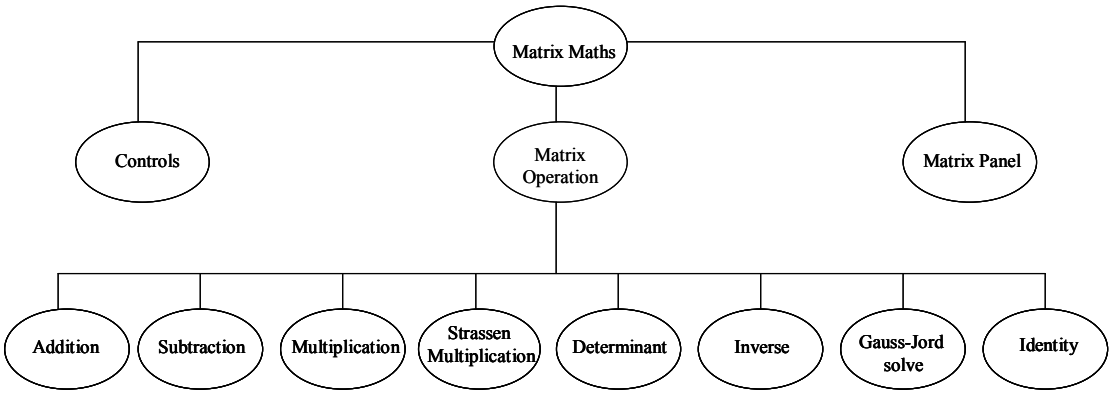


Figure4.1. Initial set of classes.

4.1. Programming Language

As listed in the requirements specification the programming language recommended to be used to implement the linear algebra system is Java. Other programming languages were disregarded as the programmer was not extremely familiar with them. Only C was taken into account as an alternative. However, certain features that Java offers and the previous experience of the programmer writing Java code inclined the balance to its side. Java is an object-oriented programming language making it ideal to implement a system for which the above classes have been identified.

Another factor that tilted the balance towards the choice of Java for the implementation of the system is the standard library that it includes for the GUI, thus less code needs to be written to develop the user interface. Other factors such as portability and security make Java the correct choice. Moreover, Java is a programming language that is very established, that has been thoroughly tested and has been proven to work appropriately.

4.2. System overview

The system is broken down into three parts or modules. These parts are the main program which executes the system, the user interface and a third module being a library containing all the mathematical functions. Figure 1 depicts the relationship between these modules.

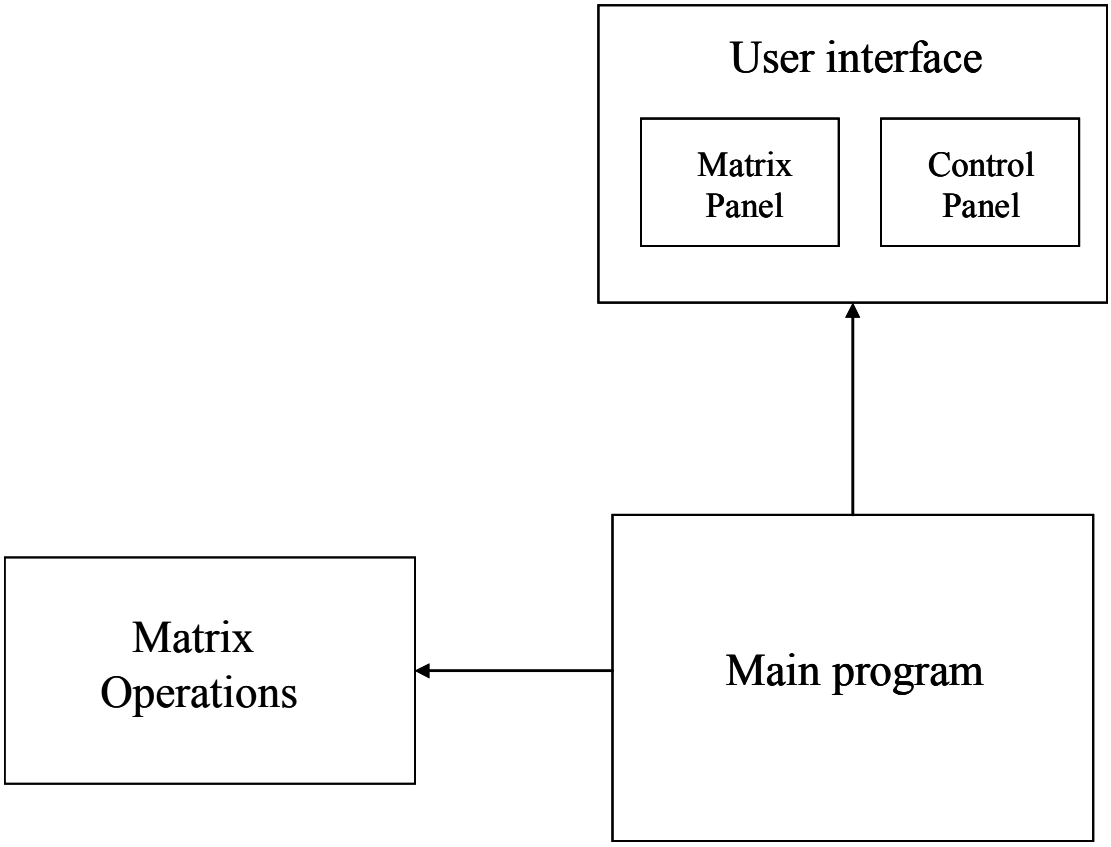


Figure4.2. An overview of the system.

The User Interface module is in turn split into two sub-modules each contained in a separate file. These two sub-modules that make up the user interface are independent from each other as they correspond to different components within the interface. The Matrix Panel contains the class that builds and handles the matrix essential for the system whereas the Control Panel contains the class that constructs the panel containing all the matrix operations available for the system.

The Matrix Operations module contains the algorithms required to perform the matrix operations, which is the entire purpose of the system i.e. a system to perform matrix computations.

The Main Program is dependant on the both modules described above. When the main program is executed it calls the classes and functions in the other two modules in order to create the control panel and matrices as well as to allow the user to perform the desired matrix computations.

4.3. Input and output considerations

The requirements specification introduced the idea of having the system to read in input data from a file as an optional feature. It has been considered that for this version of the system where the functionality is only limited it is sufficient having the user input the matrix values manually rather than having the system to read in information from a file.

Time was a running against the development of the system and it was considered that it was not worth to waste any time designing the format of the input file. Besides, the final design of the user interface does not entail command line inputs which cuts out all the tedious commands required to input data. Thus, this not being an issue it is not necessary for the system to read in files for this prototype version.

It has been decided that the maximum size for the matrices is 64 x 64. This is a totally random selection and it is thought that it is large enough for this prototype. Although it is obvious that such large matrices will not be inputted. As inputs will not be read in from a file it is unwise to allow the user to input 100 x 100 matrices, for example, manually. This would be incredibly tedious and unpractical. Ideally larger matrices should be able to be inputted to make the use of algorithms such as Strassen's worthwhile. Applying this algorithm on relatively small matrices will not differ in the execution time in comparison to the naïve method.

4.4. Data-flow models

Data-flow models are rather useful for the design of this linear algebra system. It is fairly simple to follow the flow of the data inputted by the user into the system and illustrating how the system reacts to this data. The path of the data is identified which will serve as a basis to implement the different functions appropriately. Please refer to the appendix for the data-flow models generated.

4.5. Data Structures

The data structures that will be used to store the matrices will be 2D arrays. This type of data structure is suitable for this purpose as it has two indexes that would correspond to the rows and columns of a matrix. Java stores 2D arrays as arrays of arrays, i.e. a 1D array with an index to another 1D array. The data type stored in the arrays will be doubles allowing the user to input numbers up to two decimal places.

4.6. Detailed design

It is necessary at this stage to go into more detail of some of the algorithms that are going to be included to perform the matrix operations as these may be slightly complex to code. The algorithms studied into more depth are the ones to perform multiplication by Strassen's method, calculating the determinant and the algorithm to perform the Gauss-Jordan elimination. Others like addition, subtraction or the naïve algorithm for multiplication, are trivial and thus do not need to be examined into depth before being implemented.

The mathematical theory behind these calculation methods have been explained in Section 2. Using this theory the algorithms have been constructed as efficiently as possible. The algorithms mentioned are outlined with the use of pseudo code, which will be very useful for when it comes to the implementation stage.

Strassen's algorithm

```
strassens(a, b)
a, b -> matrices

        if matrix dimension is 1x1
                return a * b

        split matrix A into 4 square matrices
        padding A with zeroes up to nearest multiple of 2
        | a1 a3 |
        | a2 a4 |

        split matrix B into 4 square matrices
        padding B with zeroes up to nearest multiple of 2
        | b1 b3 |
        | b2 b4 |

        p1 = strassens(a1 + a4, b1 + b4)
        p2 = strassens(a2 + a4, b1)
        p3 = strassens(a1, b3 - b4)
        p4 = strassens(a4, b2 - b1)
        p5 = strassens(a1 + a3, b4)
        p6 = strassens(a2 - a1, b1 + b3)
        p7 = strassens(a3 - a4, b2 + b4)

        c1 = p1 + p4 - p5 + p7
        c2 = p2 + p4
        c3 = p3 + p5
        c4 = p1 + p3 - p2 + p6

        create matrix C, same dimensions as A
        join matrices c1 to c4, remove padding added earlier
        C = | c1 c3 |
            | c2 c4 |

        return C
```

Gauss-Jordan algorithm to solve Ax = b

```
gaussjordan(a)
a -> augmented matrix

        det = 0

        for each row i
                /pivoting
                choose row p with smallest value in a[p][i]
                pivot = a[p][i]

                if pivot < 1E-10
                        matrix is singular
                        return 0
```

```
                swap row i and row p

                /determinant
                det = det * pivot
                negate det if i != p

                /row elimination
                normalise row i by dividing each element by pivot

                for each j in [i+1 .. rows(a))
                        row[j] = row[j] - a[j][i] * row[i]

        /back propagation
        for each row i, reverse order
                for each j in [0..i)
                        row[j] = row[j] - a[j][i] * row[i];

        return a, det



solve(A, b)
A, b -> matrix

        create augmented matrix
        aug = | A b |
        aug, det = gaussjordan(aug)

        if det == 0
                exception, singular matrix

        extract solution from last column of augmented matrix
        return solution
```

Determinant

```
det(A)
A -> matrix
        A, det = gaussjordan(A)
        return det
```

4.7. User Interface

The requirements document specifies that the system should implement a graphical user interface (GUI) in detriment to a console program. In current times, all the programs in the market make use of graphical interfaces. Sometimes the GUIs are given more importance than to the actual functions of the system. For the linear algebra system being developed in this project this will not be the case, and only a simple GUI will be implemented. The system must be as user-friendly as possible and it is believed that this can be attained with the use of a simple and approachable GUI.

The main issues that arose when creating the initial sketches of the graphical interface were whether to have a menu showing the options that the system offered and how to illustrate the matrices constructed. Analysis of different ideas resulted in the

development of various paper–based prototypes. None of these prototypes were convincing enough as they appeared not to be sufficiently user-friendly.

After the first few sketches and having identified the problems with them it was decided that it would be easier for the user to interact with a system that only consisted of one main window and that all the interaction took place within this window without having to refer to any menus.

The final interface design evolved solely about the core purpose of the system, i.e. taking in data, processing it and returning a result, and how to fit this into one window.

It is important that any error message introduced to the user is not intrusive and that a formal language is used to communicate the error. The message should not be abusive or offensive for the user. Moreover it should be polite and understandable as well as indicating the user what to do next.

5. Implementation

Having completed the design process, the following step is to convert all the issues discussed and explained in the previous sections into something tangible. The design aspects have contributed to structuring the implementation of the system.

The implementation of the system is split into four parts. These are then run under a main method to produce the linear algebra system. The coding has been split up in this way mainly for simplicity. This approach is also favourable in case any modifications need to be made in the future, such as adding on functionality or altering the user interface. The four files that make up the system are:

- `Matrix.java`: Contains all the algorithms to perform the matrix operations.
- `MatrixPanel.java`: Contains the methods to construct the matrix editor window.
- `ControlPanel.java`: Contains the methods to construct the controls window.
- `MatrixMaths.java`: Contains the `main()` method.

5.1. The Graphical User Interface

The graphical user interface implemented is very simple. The programmer has benefited from the libraries that Java includes to implement the graphical interface and has made use of the Java Swing components. These are located in the package `javax.swing`. Some examples below:

```
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JLabel;
import javax.swing.JTable;
```

"A graphical user interface is made up of components. Components are things like windows, buttons, checkboxes, menus, scroll bars, text fields, labels […]"[15]

The GUI implemented is an event-driven user interface. The application requires the interaction of the user, so when the user performs an action an event is taking place in the interface. Therefore, other than the component objects the GUI also has need of event and listener objects. The listener objects waits for the events to occur causing the reaction to the user's action.

An example will make the process more evident. Say we have a `JButton` component, as it name describes is a button in the interface. This button generates an event when it is pushed with the mouse by the user. The listener object will pick up this event and respond to it carrying out the function that the button describes.

[15] Niño, J., Hosch F. A., (2002, p.468)

The GUI implements a control panel which contains the buttons to perform the matrix operations. Each and every button in the control panel generates an action event which is picked up by the `ActionListener` which calls the corresponding method to perform the operation chosen by the user. The following extract from the source code relates to the `ControlPanel` constructor that sets a large default layout. `addControl` creates a new button in the panel and adds the listener to it.

```
class ControlPanel extends JPanel
{
    public ControlPanel()
    {
        super(new GridLayout(2,8));
    }

    void addControl(String name, ActionListener a)
    {
        JButton b = new JButton(name);
        add(b);
        b.addActionListener(a);
    }
}
```

The method chosen to represent the matrices in the GUI is using tables. Each matrix, the two input matrices and the output matrix, is represented by a table. This approach is appropriate as a matrix is composed of rows and columns as are tables.

The matrices are made up of a panel, `JPanel`, which contains the table, `JTable`, The table is attached to a scroll pane, `JScrollPane`. Another panel is contained within the main panel which holds the matrix size controls which are created by `JSpinner`.

DEFINITION: A single line input field that lets the user select a number or an object value from an ordered sequence. Spinners typically provide a pair of tiny arrow buttons for stepping through the elements of the sequence. The keyboard up/down arrow keys also cycle through the elements. The user may also be allowed to type a (legal) value directly into the spinner.[16]

The class `MatrixPanel` builds the matrix windows. These are editable in order to obtain different sized matrices. The `MatrixPanel` constructor creates a `JTable` and two `JSpinners`. The class `MatrixModel` is an adaptor used by `JTable` to edit a matrix.

[16] http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JSpinner.html

```
    public class MatrixPanel extends JPanel
{
    class MatrixModel extends AbstractTableModel
    {

                        ⋮
                        ⋮
                        ⋮
                        ⋮
}

    class MatrixChangeListener implements ChangeListener
    {
        private ChangeListener cl;
        MatrixChangeListener(ChangeListener ch) { cl = ch; }
        public void stateChanged(ChangeEvent e) {
cl.stateChanged(e); }
    }

                        ⋮

    public MatrixPanel(Matrix m)
    {

    }
```

The class `MatrixChangeListener` listens for the changes from the `JSpinners` and passes the message on the change to `MatrixMaths`, this being the main() method.

The `addListener` method creates the `MatrixChangeListener` to listen to the `JSpinners`. The `updateMatrix` method will update the table with the new matrix and turn off the `MatrixChangeListener` so the `JSpinners` can be updated.

```
public void addListener(ChangeListener cl)
    {
                ⋮
    }

public void updateMatrix(Matrix m)
    {
                ⋮
    }
```

5.2. The `main()` method

```
public static void main(String[] args)
      {
          MatrixMaths mm = new MatrixMaths();
      }
```

Running the program executes the main method extracted above. When the `main()` is executed a `MatrixMaths` object is created. The main function calls the methods found in `ControlPanel.java` and `MatrixPanel`.java to construct the interface.

The control panel is built and the buttons are added to it. When the buttons are pushed they call the operation methods found in Matrix.java.

```
        // Create the control panel
        controls = new JFrame("Controls");
        ControlPanel controlpanel = new ControlPanel();
        controls.setContentPane(controlpanel);

controls.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Anonymous classes with operations code
        controlpanel.addControl("A + B", new MatrixAdaptor()
{
            public void action() throws MatrixException { mC
= mA.plus(mB); }
        });

                                ⋮
                                ⋮
```

```
        controlpanel.addControl("A * B (Strassens)", new
MatrixAdaptor() {
            public void action() throws MatrixException {
mC = mA.multstrass(mB); }
        });
                                ⋮
                                ⋮
```

The matrices are also created; these being initially 4 x 4 zero matrices. The matrices are built within individual windows as explained earlier and are assigned a position on the screen.

It is important that the windows have an attractive appearance to eye. The following command applies a default set of decorations on the windows.

```
JFrame.setDefaultLookAndFeelDecorated(true);
```

The main method implements an ActionListener as well as a ChangeListener. The action listener as mentioned previously "listens" for any events that generated, however, the change listener "listens" to any changes in any event already picked up. This will be the case when the values of the matrix elements are being inputted and the matrix sizes are being altered. Once this change has been picked up the new input will be updated making the matrix valid with the new data.

```
class MatrixAdaptor implements ActionListener
    {
                ⋮

class MatrixChangeListener implements ChangeListener
    {
                ⋮
```

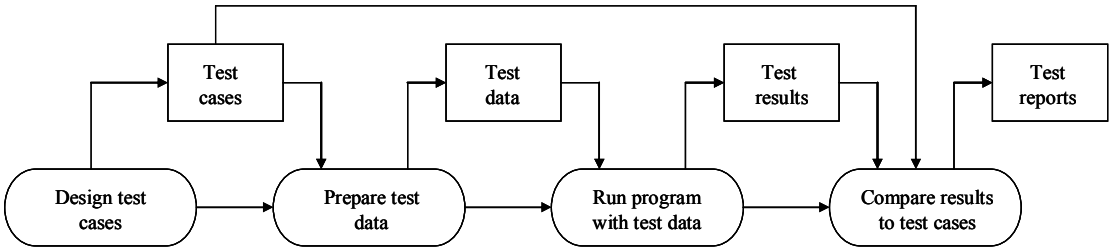
Once all the windows have been built they have to be displayed on the screen. The following extract is responsible for this.

```
controls.pack();
controls.setVisible(true);
frameA.pack();
frameA.setVisible(true);
frameB.pack();
frameB.setVisible(true);
frameC.pack();
frameC.setVisible(true);
```

6. Testing

The testing process of the system is an important procedure that will allow the software developer to validate the requirements specification and to verify the design.

"Software testing is a process used to identify the correctness, completeness and quality of developed computer software."[17] Testing procedures are used to find any flaws in the software, in other words the testing procedures will find defects in the system but it will not assure the non-existence of them.



6.1. Objective of testing

In essence, the objective of the software testing procedure is to ensure that the system delivers the functionality required as specified in the requirements document and that the software is of sufficient quality to fulfil user expectations.

6.2. Testing scope

Several types of testing procedures may be put into practice to test a system. However, only a selection of test types have been chosen to be carried out upon this system mainly due to time constraints, as a test process may extend throughout a long period of time.

For this system the importance evolves about the functionality and the usability. So the principal aim of the testing procedures on this system is to prove that the system performs correctly the mathematical operations on matrices. In addition, it also needed to prove that the users are able to perform these operations in a simple and straightforward manner.

To test for the functionality of the system black-box tests will be performed and to test for the usability of the system the users will interact with the system.

[17] Somerville, I, (1982 p.441)

6.3. Functional/Black-box testing

DEFINITION: Black box testing or functional testing is used in computer programming, software engineering and software testing to check that the outputs of a program, given certain inputs, conform to the functional specification of the program.[18]

Both terms, functional and black-box, are used indifferently although the latter is the more convenient description as the name explains the process. With this type of testing procedure the system is like a 'black-box' where the code is not looked at and the validity of the system is determined solely by the output results to the given inputs.

There are never enough test cases to test a system, these are potentially endless, but with enough test cases applied to a system it can be ensured that the system meets the functionality expected. By applying inputs and analysing the corresponding outputs the functionality of the system is being tested. If the output value does not correspond to the expected output a flaw has been identified. The diagram below illustrates a black-box test model where I_e are inputs with high probability of producing an error.

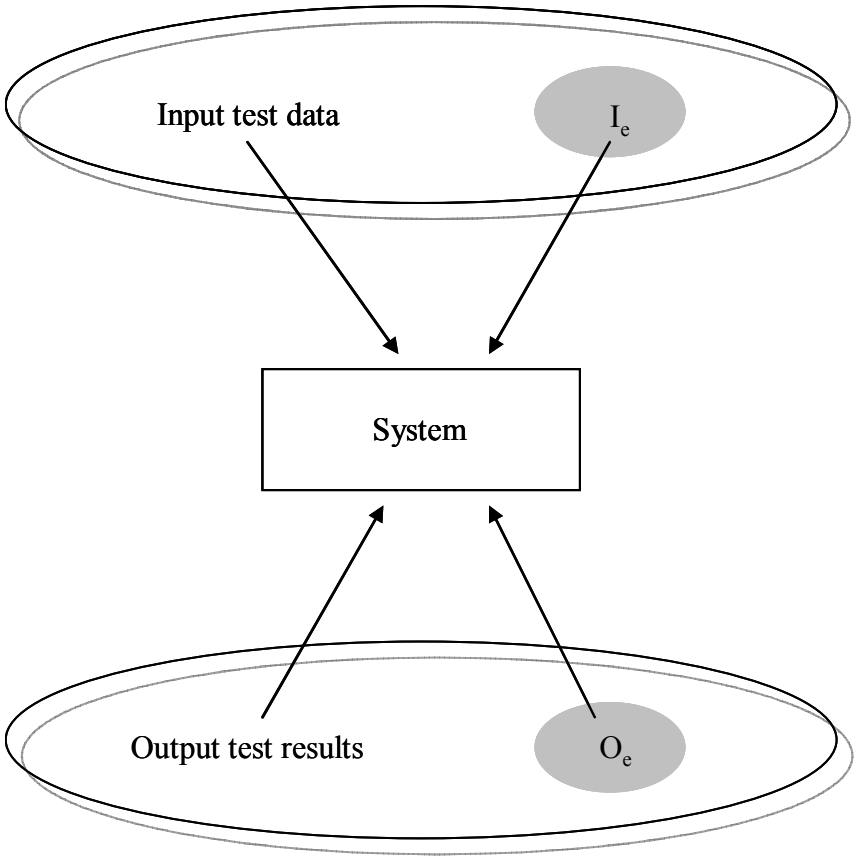


Figure1. Illustration emulating black-box test.[19]

[18] Definition from Wikipedia On-line encyclopedia

[19] Somerville, I. (1982 p.444)

There are various approaches within black-box testing, one being equivalence partitioning. This is a method to reduce the potentially infinite number of test cases that can be applied to a system. Equivalence partitioning, in essence, divides the test cases into groups of similar description as it is reasonable to believe that the system will react in a similar to similar kinds of input data. Once the input data has been classified into the different partitions the test cases are selected from these groups which reduces considerably the total number of test cases. One method of selecting the test cases to be applied to the system is by choosing boundary values within the partitions. Usually systems react abnormally to input data that are at extreme ends of the input range, so choosing boundary values will increase the chances of detecting any anomalies.

A carefully designed test plan will be drawn up for the linear algebra system so as to detect as many flaws in the functions of the system. These are described in the following table.

6.4. Usability testing

Usability testing as its own term explains tests the usability of the system developed. The best approach to this is to get users totally unfamiliar with the system itself to interact with it. It would help if the users are familiar with the purpose of the system so that they can expect certain behaviours. With respect to the linear algebra system the users should have a mathematical background so that they are familiar with handling matrices and their operations.

Specific test scenarios will be designed for the users to go through while their actions are being annotated by the tester. The users will also be allowed to interact with the system freely. Having completed the tests the users will be asked to comment on issues that particularly caught their attention and either liked or disliked. Any suggestions made by the user will be taken into consideration and analysed and perhaps introduced to the system before delivery or left to future updates.

What is being tested?

The usability tests are mainly carried out to approve the user interface put into use by the system. The interaction of the users with the system will return sufficient feedback to be able to judge whether the user interface is satisfactory and meets the user requirements specified.

At the same time the functionality of the system is being examined and any errors that may have not been caught in the black-box tests may appear as the users will be carrying out matrix operations.

Test Scenarios

A number of 'test' users, yet to be decided, will be required to carry out three tasks per user. The users will have access to a paper-based user manual that they make use of at any time during the testing.

Task1

Run the program and perform matrix addition on two matrices of any size.

Task 2

Run the program and perform matrix subtraction on two matrices of any size.

Task 3

Run the program and perform matrix multiplication on two matrices of any size.

Task 4

Run the program and perform matrix multiplication using Strassen's algorithm on two matrices of any size.

Task 5

Run the program and solve a system of linear equations.

Task 6

Run the program and find the determinant of a 5 x 5 matrix.

Task 7

Run the program and find the inverse of a 4 x 4 matrix.

Task 8

Run the program and construct a 10 x 10 identity matrix.

Task 9

Run the program and perform a matrix addition of any size, then use the result matrix to carry out another addition.

Task 10

Run the program and perform a matrix subtraction of any size, then use the result matrix to carry out addition of the result matrix with another matrix of your choice.

Task 11

Run the program and add two 4 x 4 matrices, then calculate the determinant of the result.

Task 12

Run the program and construct two matrices, A and B, of any desired size and calculate AB and BA.

Task 13

Run the program and construct matrix A, 3 x 3, then proceed to find A^{-1} and finally make sure that this is the correct answer by verifying that AA^{-1} = Identity matrix.

6.5. Test Results Evaluation

Please refer to the appendix where you will find the test results. It may be concluded after evaluating both black-box and usability test results that the system delivers functionality set at the beginning of the project.

The tests have been design specifically in accordance to the requirement specification with the aim of validating these. Therefore, it can be said that testing process has been successful and that the system is valid as part of this project.

It must be observed that the system did not pass one of the test cases. The test case #6 was not able to be completed, thus resulting in a test failure. The particular functionality that was being tested was looked upon in detail in the source code in order to amend the error. The source code was fixed and the test was run again. The second attempt resulted in a positive outcome and the expected result was met.

It is understood that the system is valid for the purposes of this project. It would have been desirable to test the system further but time was pushing and the delivery date close so only the minimum test requirements were carried out. Furthermore, it is evident that there are various other important tests that a software program needs to go through in a commercial software development project.

7. Conclusion

The system conforms to the requirement specification listed when commencing with the development of the system. The linear algebra system was accepted positively by users who carried out the usability tests. Thus the overall evaluation to make is a favourable one.

However, taking a closer look at all the stages and aspects that have been involved for the development of this system there is both positive and negative critique.

If this project was to be carried out once again several aspects should be considered more closely from day 1. It is of great importance planning the time more accurately by being more realistic with the time allocations. The flaw in the planning skills and the lack of commitment to keep to the plan at specific points has become evident because certain areas have only met the minimum requirements due to the lack of time.

Performing a linear analysis of the project lets start from the beginning and discuss the way through the project. It must be remarked that the literature review had a very positive outcome as it put the development of the project on the right track from the very start. A good comprehension of the mathematics that evolves around this project was crucial and this was achieved positively.

Moving on, the next step was specifying the requirements for the system. This process was significantly more demanding than expected at first sight. At this point is where the time plan began to fall apart. It was extremely complicated to narrow down the scope of the system as it is potentially infinite which resulted in spending more time at this particular area than expected.

The design and implementation sections are the less dense parts of the project. This is due to the nature of the system being implemented. The major design issues evolved around the interface and the usability. The algorithms implemented were not complicated therefore not requiring to be specifically designed before implementation. The implementation was kept fairly simple making the source code almost self-explanatory with the aid of the comments introduced.

It is true that from the test results obtained from the testing procedures the system is valid for submission as it meets the requirement specification. Ideally, the system should have been tested more intensely and other forms of tests should have been applied apart from the tests undertaken.

However, at this stage the main concern was whether such as system would be accepted positively by the target users. As a matter of fact, the system was accepted very positively by users that tried it. Moreover, they were keen on using such a system in the near future. This simple computer linear algebra system should be set as the first step to producing a system specialized on performing matrix related calculations.

7.1. Future work

It is fairly evident that the functionality of the linear algebra system developed is very restricted and has been kept to the basics. There are many other functions that may be incorporated to the system in the future to make it a more complete system. These being for example:

- Rank calculation
- Trace calculation
- Vector spaces of matrices
- Projection matrices
- etc. etc.

Other areas that may be worked on are the user interface, although this will come as a consequence of increasing the functionality of the system. The interface will have to incorporate new features in order to allow the user to benefit of any updates in the functionality. As some of the users that tested the system suggested it may be a good idea to integrate the user guide into the system so that it can be accessed while performing any operations.

Depending on the acceptance of an enhanced version of the system it may be worth considering making the system available on-line and expanding the target user range. Not limiting the system to A-level or undergraduate users. Of course, if this were the case the whole requirement specification has to be redefined as the scope is much wider.

7.2. Personal analysis

The goal set at the beginning of this project was to develop a system that could take matrices and perform different algebraic operations upon them. I am pleased to say that this has been achieved despite the doubts I had at the start.

The subject being consider is very extensive and also potentially infinite. Maths and linear algebra is a subject that I enjoy. When it came to selecting the functions to be included in the system I found it extremely difficult to narrow them down to suitable number that would make it possible to implement the system within the time limit that we assigned.

The extensive reading and investigation accomplished in the literature review introduced me to certain mathematical aspects that I was unaware of and helped me understand them. This part of the project paid off extremely well for the further development of the final system. The literature review also put on me course in the requirements analysis section. Having attained a thorough comprehension of the functions and algorithms to be used eased the process of defining the requirements of the system. However, it was not sufficient with the literature review and the analysis of existing products and walk-through scenario use-cases helped to pull out less evident ones.

The design process was somewhat trickier than the previous sections mentioned as I found it difficult to apply to this system the different design models that I found in the literature. It was important to get the design documents straight in order to ease the implementation of the system. I have been obliged to perfection my programming skills and overcome my dissatisfaction to this area of Computer Science. However, my dislike to programming has decreased considerably having completed this system.

I am very much satisfied with the effort put into this project. I have attained many important skills, and polished others, that I will be able to apply in my daily life, such as, time management and planning.

8. Bibliography

COCKBURN, A., 2001. *Writing Effective Use Cases*. Addison-Wesley.

CORMEN, T. H., LEISERSON, C.E., RIVEST, R.L., STEIN, C., 1990. *Introduction to Algorithms*. 2nd Ed. Massachusetts: MIT Press.

NIÑO, J. HOSCH, F. A., 2002. *An introduction to programmming and object oriented design using Java*. Wiley & Sons.

KRONSJÖ, L., 1987. *Algorithms: Their Complexity and Efficiency*. 2nd Ed. Wiley & Sons.

WILLIAMS, G., 1937. *Linear Algebra with Applications*. 4th Ed. Jones and Bartlett.

SOMMERVILLE, I., 1982. *Software Engineering*.6th Ed. Addison-Wesley.

STROTMANN, A., 1995. *The REDUCE Computer Algebra System*. University of Köln. Available from: http://www.uni-koeln.de/REDUCE/#intro.

MAPLE 8. User Manual – Introduction.

1.6. Further Reading

CHAMBERT, J., 1998. *A History of Algorithms: From the Pebble to the Microchip.* Springer.

DROMEY, R.G., 1982. *How to Solve it by Computer*. London: Prentice-Hall International.

GROSSMAN, S. I., 1994. *Elementary Linear Algebra*. 5th. Sauders College Publishing.

McCONNELL, J.J., 2001. *Analysis of Algorithms: An Active Learning Approach.* Massachusetts: Jones and Bartlett.

NERING, E.D., 1970. *Linear Algebra and Matrix Theory*. 2nd. Wiley & Sons.

REDMOND-PYLE, D., MOORE, A., 1995. *Graphical User Interface Design and Evaluation*. Oxford: Prentice Hall.

SHNEIDERMAN, B., 1998. *Designing the User Interface*. Addison-Wesley.

THORPE, J.A., KUMPEL, P.G., 1984. *Elementary Linear Algebra*. Sauders College Publishing.

Prof. Arsham H., 1994. *Unification of System of Linear Equations, Matrix Inversion, and Linear Programming*. Available from: http://home.ubalt.edu/ntsbarsh/opre640a/partXII.htm.

Van der Zanden, B., *Analysis of Algorithms and Selection of Algorithms*. Available from: http://www.cs.utk.edu/~bvz/bvz/classes/cs302/notes/complexity.html.

Carrano, *Algorithm Efficiency*. Available from: http://homepages.ius.edu/gmanwani/C202/algorithm%20efficiency.pdf.

9. Appendices

A1. Requirements appendix
- Questionnaires

A2. Design appendix
- Data-flow models

A3. Implementation appendix
- Source code

A4. Testing appendix
- Black-box test results
- Usability test results

A5. User Guide

A1 Requirements Appendix

A2 Design Appendix

A3 Implementation Appendix

```
//*********************************************//
// MatrixMaths.java: Contains the main() method. //
// Builds the interface and displays it and      //
// performs the matrix operations.          //
//*********************************************//



import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class MatrixMaths
{
    Matrix mA, mB, mC;
    MatrixPanel panelA, panelB, panelC;
    JFrame controls;

    class MatrixAdaptor implements ActionListener
    {
        public void action() throws MatrixException {}
        public void actionPerformed(ActionEvent e)
        {
            try {
                //Try action
                action();
                panelA.updateMatrix(mA);
                panelB.updateMatrix(mB);
                panelC.updateMatrix(mC);
            }
            catch(MatrixException err) {

                //Show error message
                JOptionPane.showMessageDialog(controls,
err.error);
            }
        }
    }

    class MatrixChangeListener implements ChangeListener
    {
        private int which;
        MatrixChangeListener(int i) { which = i; }

        public void stateChanged(ChangeEvent e)
        {
            if(which == 0)
                mA = newMatrixSize(mA, panelA);
```

```
            else
                mB = newMatrixSize(mB, panelB);
        }

        public Matrix newMatrixSize(Matrix m, MatrixPanel p)
        {
            if(m.rows() != p.getMatrixRows() || m.columns() !=
p.getMatrixCols())
                m = new Matrix(p.getMatrixRows(),
p.getMatrixCols());

            p.updateMatrix(m);
            return m;
        }
    }


    public MatrixMaths()
    {
        // Make sure we have nice window decorations
        JFrame.setDefaultLookAndFeelDecorated(true);

        // Create the control panel
        controls = new JFrame("Controls");
        ControlPanel controlpanel = new ControlPanel();
        controls.setContentPane(controlpanel);

controls.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Anonymous classes with operations code
        controlpanel.addControl("A + B", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.plus(mB); }
        });
        controlpanel.addControl("A - B", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.minus(mB); }
        });
        controlpanel.addControl("A * B", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.mult(mB); }
        });
        controlpanel.addControl("A * B (Strassens)", new
MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.multstrass(mB); }
        });
        controlpanel.addControl("Solve Ax = B", new
MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.solve(mB); }
        });
        controlpanel.addControl("Inv A", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mA.inverse(); }
        });
```

```
        controlpanel.addControl("Inv B", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
mB.inverse(); }
        });
        controlpanel.addControl("Det A", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
new Matrix(1,1); mC.set(0, 0, mA.det()); }
        });
        controlpanel.addControl("Det B", new MatrixAdaptor() {
            public void action() throws MatrixException { mC =
new Matrix(1,1); mC.set(0, 0, mB.det()); }
        });
        controlpanel.addControl("A = Ident", new
MatrixAdaptor() {
            public void action() throws MatrixException { mA =
Matrix.identity(mA.rows()); }
        });
        controlpanel.addControl("B = Ident", new
MatrixAdaptor() {
            public void action() throws MatrixException { mB =
Matrix.identity(mB.rows()); }
        });
        controlpanel.addControl("Swap A&B", new
MatrixAdaptor() {
            public void action() throws MatrixException {
Matrix m = mA; mA = mB; mB = m; }
        });
        controlpanel.addControl("A = C", new MatrixAdaptor() {
            public void action() throws MatrixException { mA =
new Matrix(mC); }
        });
        controlpanel.addControl("B = C", new MatrixAdaptor() {
            public void action() throws MatrixException { mB =
new Matrix(mC); }
        });

        // Create and set up the matrix windows
        JFrame frameA = new JFrame("Matrix A");
        JFrame frameB = new JFrame("Matrix B");
        JFrame frameC = new JFrame("Matrix Result");

        frameA.setLocation(100,100);
        frameB.setLocation(250,250);
        frameC.setLocation(400,400);
        frameA.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frameB.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frameC.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set up matrices
        mA = new Matrix(4, 4);
        mB = new Matrix(4, 4);
        mC = new Matrix(4, 4);

        // Create and set up the content panes
        panelA = new MatrixPanel(mA);
        panelB = new MatrixPanel(mB);
```

```
        panelC = new MatrixPanel(mC);

        panelA.addListener(new MatrixChangeListener(0));
        panelB.addListener(new MatrixChangeListener(1));

        frameA.setContentPane(panelA);
        frameB.setContentPane(panelB);
        frameC.setContentPane(panelC);

        // Display the window.
        controls.pack();
        controls.setVisible(true);
        frameA.pack();
        frameA.setVisible(true);
        frameB.pack();
        frameB.setVisible(true);
        frameC.pack();
        frameC.setVisible(true);
    }

     public static void main(String[] args)
     {
         MatrixMaths mm = new MatrixMaths();
     }
}




//*********************************************//
// Matrix.java: Contains algorithms to perfrom   //
// the matrix operations.              //
//                                  //
//*********************************************//



class Matrix
{
     private int m, n;           // number of rows, columns
     private double[][] x;       // matrix

     // Construct a matrix, all zeros
     public Matrix(int rows, int cols)
     {
          m = rows;
          n = cols;
          x = new double[m][n];
     }

     // Construct a matrix from a 2d array
     public Matrix(double[][] a)
     {
          m = a.length;
```

```
            n = a[0].length;
            x = new double[m][n];

            for(int i = 0; i < m; i++)
                  System.arraycopy(a[i], 0, x[i], 0, n);
      }

      // Construct a copy of a Matrix
      public Matrix(Matrix o)
      {
            m = o.x.length;
            n = o.x[0].length;
            x = new double[m][n];

            for(int i = 0; i < m; i++)
                  System.arraycopy(o.x[i], 0, x[i], 0, n);
      }

      // Accessors
      public int rows() { return m; }
      public int columns() { return n; }
      public double get(int i, int j) { return x[i][j]; }
      public void set(int i, int j, double n) { x[i][j] = n; }

      // Add two matrices, element by element
      public Matrix plus(Matrix o) throws MatrixException
      {
            if(m != o.m || n != o.n)
                  throw new MatrixException("Matrix dimensions
do not match");

            Matrix r = new Matrix(m, n);

            for(int i = 0; i < m; i++)
                  for(int j = 0; j < n; j++)
                        r.x[i][j] = x[i][j] + o.x[i][j];

            return r;
      }

      // Subtract two matrices, element by element
      public Matrix minus(Matrix o) throws MatrixException
      {
            if(m != o.m || n != o.n)
                  throw new MatrixException("Matrix dimensions
do not match");

            Matrix r = new Matrix(m, n);

            for(int i = 0; i < m; i++)
                  for(int j = 0; j < n; j++)
                        r.x[i][j] = x[i][j] - o.x[i][j];

            return r;
      }
```

```
	// Matrix multiply, naive algorithm
	public Matrix mult(Matrix o) throws MatrixException
	{
		if(n != o.m)
			throw new MatrixException("Matrix dimensions
do not match");

		Matrix r = new Matrix(m, o.n);

		for(int i = 0; i < o.n; i++)
		{
			for(int j = 0; j < m; j++)
			{
				double s = 0.0;
				for(int k = 0; k < n; k++)
					s += x[j][k] * o.x[k][i];
				r.x[j][i] = s;
			}
		}

		return r;
	}

	// Matrix multiply, Strassen's algorithm
	// Square matrices only
	public Matrix multstrass(Matrix o) throws MatrixException
	{
		if(m != n || o.m != o.n)
			throw new MatrixException("Matrices must be
square");
		if(m != o.m)
			throw new MatrixException("Matrix dimensions
do not match");

		Matrix r = new Matrix(n, n);

		if(n == 1)
		{
			r.x[0][0] = x[0][0] * o.x[0][0];
		}
		else
		{
			int n0 = (n + 1) / 2, n1 = n - n0;

			Matrix a1 = new Matrix(n0, n0), a2 = new
Matrix(n0, n0), a3 = new Matrix(n0, n0), a4 = new Matrix(n0,
n0);
			Matrix b1 = new Matrix(n0, n0), b2 = new
Matrix(n0, n0), b3 = new Matrix(n0, n0), b4 = new Matrix(n0,
n0);

			for(int i = 0; i < n0; i++)
			{
				System.arraycopy(x[i], 0, a1.x[i], 0,
n0);
```

```
                        System.arraycopy(x[i], n0, a3.x[i], 0,
n1);
                        System.arraycopy(o.x[i], 0, b1.x[i], 0,
n0);
                        System.arraycopy(o.x[i], n0, b3.x[i], 0,
n1);
                  }

                  for(int i = 0; i < n1; i++)
                  {
                        System.arraycopy(x[i+n0], 0, a2.x[i], 0,
n0);
                        System.arraycopy(x[i+n0], n0, a4.x[i], 0,
n1);
                        System.arraycopy(o.x[i+n0], 0, b2.x[i],
0, n0);
                        System.arraycopy(o.x[i+n0], n0, b4.x[i],
0, n1);
                  }

                  Matrix p1 =
(a1.plus(a4)).multstrass(b1.plus(b4));
                  Matrix p2 = (a2.plus(a4)).multstrass(b1);
                  Matrix p3 = a1.multstrass(b3.minus(b4));
                  Matrix p4 = a4.multstrass(b2.minus(b1));
                  Matrix p5 = (a1.plus(a3)).multstrass(b4);
                  Matrix p6 =
(a2.minus(a1)).multstrass(b1.plus(b3));
                  Matrix p7 =
(a3.minus(a4)).multstrass(b2.plus(b4));

                  a1 = p1.plus(p4).minus(p5).plus(p7);
                  a2 = p2.plus(p4);
                  a3 = p3.plus(p5);
                  a4 = p1.plus(p3).minus(p2).plus(p6);

                  for(int i = 0; i < n0; i++)
                  {
                        System.arraycopy(a1.x[i], 0, r.x[i], 0,
n0);
                        System.arraycopy(a3.x[i], 0, r.x[i], n0,
n1);
                  }

                  for(int i = 0; i < n1; i++)
                  {
                        System.arraycopy(a2.x[i], 0, r.x[i+n0],
0, n0);
                        System.arraycopy(a4.x[i], 0, r.x[i+n0],
n0, n1);
                  }
            }

            return r;
      }
```

```
	// Matrix inverse
	public Matrix inverse() throws MatrixException
	{
		if(m != n)
			throw new MatrixException("Matrix must be
square");

		double[][] A = new double[m][n*2];
		double[][] Ainv = new double[m][n];

		for(int i = 0; i < m; i++)
		{
			System.arraycopy(x[i], 0, A[i], 0, n);
			A[i][n+i] = 1.0;
		}

		if(gaussjordan(A) == 0.0)
			throw new MatrixException("Matrix is
singular");

		for(int i = 0; i < m; i++)
			System.arraycopy(A[i], n, Ainv[i], 0, n);

		return new Matrix(Ainv);
	}

	// Gauss-Jordan solver, solution of Mx = b
	public Matrix solve(Matrix b) throws MatrixException
	{
		if(m != n)
			throw new MatrixException("Matrix must be
square");
		if(m != b.m || b.n != 1)
			throw new MatrixException("Vector dimension
does not match matrix");

		double[][] A = new double[m][n+1];
		Matrix r = new Matrix(m, 1);

		for(int i = 0; i < m; i++)
		{
			System.arraycopy(x[i], 0, A[i], 0, n);
			A[i][n] = b.x[i][0];
		}

		if(gaussjordan(A) == 0.0)
			throw new MatrixException("Matrix is
singular");

		for(int i = 0; i < m; i++)
			r.x[i][0] = A[i][n];

		return r;
	}

	// Matrix determinant, square matrices only
```

```
	public double det() throws MatrixException
	{
		if(m != n)
			throw new MatrixException("Matrix must be
square");

		double[][] A = new double[m][n];

		for(int i = 0; i < m; i++)
			System.arraycopy(x[i], 0, A[i], 0, n);

		return gaussjordan(A);
	}

	// Gauss-Jordan algorithm on augmented matrix A
	// returns determinant, A is modified
	private double gaussjordan(double[][] A)
	{
		int Acols = A[0].length;
		double det = 1.0;

		// Gauss-Jordan
		for(int k = 0; k < m; k++)
		{
			// Choose pivot
			int p = k;
			double pivot = 0.0;
			for(int i = k; i < m; i++)
			{
				if(Math.abs(pivot) < Math.abs(A[i][k]))
				{
					p = i;
					pivot = A[i][k];
				}
			}

			// Singular matrix check
			if(Math.abs(pivot) < 1.0e-10)
				return 0.0;

			// Determinant calcuation
			det = det * pivot;
			if(p != k)
				det = -det;

			// Swap rows
			double[] tmpA = A[k];
			A[k] = A[p];
			A[p] = tmpA;

			// Normalise row
			for(int j = 0; j < Acols; j++)
				A[k][j] = A[k][j] / pivot;

			// Row elimination
			for(int i = k+1; i < m; i++)
```

```
                {
                        double scale = A[i][k];
                        for(int j = 0; j < Acols; j++)
                                A[i][j] = A[i][j] - scale *
A[k][j];
                }
        }

        // Back propagation
        for(int i = m-1; i >= 0; i--)
        {
                for(int j = 0; j < i; j++)
                {
                        for(int k = n; k < Acols; k++)
                                A[j][k] = A[j][k] - A[j][i] *
A[i][k];
                }
        }

        return det;
    }

    // Create an identity matrix
    public static Matrix identity(int size)
    {
        Matrix r = new Matrix(size, size);

        for(int i = 0; i < size; i++)
                r.x[i][i] = 1.0;

        return r;
    }
}

class MatrixException extends Exception
{
    public String error;

    public MatrixException(String s)
    {
        error = s;
    }
}
```

```
//*********************************************//
// MatrixPanel.java: Contains methods to biuld   //
// matrix windows.                      //
//                                 //
//*********************************************//



import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.JSpinner;
import javax.swing.SpinnerNumberModel;
import java.awt.Dimension;
import java.awt.BorderLayout;

import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class MatrixPanel extends JPanel
{
    class MatrixModel extends AbstractTableModel
    {
        private Matrix m;

        public MatrixModel(Matrix matrix) { m = matrix; }
        public void updateMatrix(Matrix matrix) { m = matrix;
fireTableStructureChanged(); }

        public int getRowCount() { return m.rows(); }
        public int getColumnCount() { return m.columns(); }
        public boolean isCellEditable(int row, int col) {
return true; }
        public Class getColumnClass(int col)
        {
            return new Double(0.0).getClass();
        }
        public Object getValueAt(int row, int col)
        {
            return new Double(m.get(row, col));
        }
        public void setValueAt(Object value, int row, int col)
        {
            m.set(row, col, ((Double)value).doubleValue());
        }
    }

    class MatrixChangeListener implements ChangeListener
    {
        private ChangeListener cl;
        MatrixChangeListener(ChangeListener ch) { cl = ch; }
```

```
        public void stateChanged(ChangeEvent e) {
cl.stateChanged(e); }
    }

    private JSpinner rows, cols;
    private JTable table;
    private MatrixModel model;

    public MatrixPanel(Matrix m)
    {
        super(new BorderLayout());

        // Create the scroll pane and add the table to it
        model = new MatrixModel(m);
        table = new JTable(model);
        JScrollPane scrollPane = new JScrollPane(table);
        table.setPreferredScrollableViewportSize(new
Dimension(50 * m.columns(), 16 * m.rows()));

        // Create the size controls
        final int MAX_SIZE = 64;
        rows = new JSpinner(new SpinnerNumberModel(m.rows(),
0, MAX_SIZE, 1));
        cols = new JSpinner(new
SpinnerNumberModel(m.columns(), 0, MAX_SIZE, 1));

        JPanel sizepanel = new JPanel();
        sizepanel.add(new JLabel("Rows"));
        sizepanel.add(rows);
        sizepanel.add(new JLabel("Columns"));
        sizepanel.add(cols);

        // Create matrix panel
        add(scrollPane, BorderLayout.CENTER);
        add(sizepanel, BorderLayout.PAGE_END);
    }

    public void addListener(ChangeListener cl)
    {
        // Listen to matrix size changes
        MatrixChangeListener listen = new
MatrixChangeListener(cl);
        rows.addChangeListener(listen);
        cols.addChangeListener(listen);
    }

    public void updateMatrix(Matrix m)
    {
        model.updateMatrix(m);
        rows.setValue(new Integer(m.rows()));
        cols.setValue(new Integer(m.columns()));
    }

    public int getMatrixRows() { return
((Integer)rows.getValue()).intValue(); }
```

```
    public int getMatrixCols() { return
((Integer)cols.getValue()).intValue(); }
}


//*********************************************//
// ControlPanel.java: Contains methods to build  //
// panel of operation controls.              //
//                                  //
//*********************************************//



import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.GridLayout;
import java.awt.event.ActionListener;

class ControlPanel extends JPanel
{
    public ControlPanel()
    {
        super(new GridLayout(2,8));
    }

    void addControl(String name, ActionListener a)
    {
        JButton b = new JButton(name);
        add(b);
        b.addActionListener(a);
    }
}
```

A4 Testing Appendix

Test data for black-box testing

$$A = \begin{pmatrix} 2 & 0 & 3 \\ 1 & 0 & 2 \\ 2 & 1 & 1 \end{pmatrix}, B = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}, C = \begin{pmatrix} 2 & 3 \\ 3 & 2 \\ 1 & 0 \end{pmatrix}, D = \begin{pmatrix} 1 & 0 & 4 & 1 & 0 \\ 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 0 & 1 \\ 1 & 3 & 1 & 0 & 2 \\ 1 & 1 & 3 & 2 & 1 \end{pmatrix}, E = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}, G = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 2 & 1 & 0 & 3 \\ 1 & 0 & 3 & 2 \\ 0 & 3 & 2 & 1 \end{pmatrix}, H = \begin{pmatrix} 4 & 0 & 3 \\ 5 & 6 & 0 \\ 1 & 2 & 0 \end{pmatrix}, J = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

Test case	Expected Result	(P)ass / (F)ail
A + H	$\begin{pmatrix} 6 & 0 & 6 \\ 6 & 6 & 2 \\ 3 & 3 & 1 \end{pmatrix}$	P
B + C	error	P
H – A	$\begin{pmatrix} 2 & 0 & 0 \\ 4 & 0 & -2 \\ -1 & 1 & -1 \end{pmatrix}$	P
G – F	Error	P
E * C	$\begin{pmatrix} 6 & 5 \\ 4 & 6 \end{pmatrix}$	P
C * E without re-entering matrix data.	$\begin{pmatrix} 8 & 2 & 2 \\ 7 & 3 & 3 \\ 1 & 1 & 1 \end{pmatrix}$	F (fixed)
B * F	error	P
D * D	$\begin{pmatrix} 10 & 7 & 13 & 1 & 6 \\ 9 & 10 & 16 & 5 & 7 \\ 8 & 5 & 17 & 5 & 4 \\ 8 & 9 & 18 & 8 & 6 \\ 11 & 12 & 17 & 4 & 9 \end{pmatrix}$	P
D * G	Error	P

Solve Ax = B	$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$	P
Solve Hx = C	Error	P
Inverse of A	$\begin{pmatrix} 2 & -3 & 0 \\ -3 & 4 & 1 \\ -1 & 2 & 0 \end{pmatrix}$	P
Inverse of J	Error	P
Determinant of A	-1	P
Determinant of F	Error	P
$I_{10\ x10}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	P
(result (A + H) + A)	$\begin{pmatrix} 8 & 0 & 9 \\ 7 & 6 & 4 \\ 5 & 4 & 2 \end{pmatrix}$	P

Usability test results

The user's carrying out the usability tests were interviewed in order to get some background information before handing them over the tasks to be completed.

User #1

Do you have a mathematical background? Have you performed matrix calculations before?

Yes, I have completed A-level maths and I have done the basic matrix calculations.

Have you used any computer program to solve matrix calculations?

No.

User #1 was requested to complete tasks 1, 4 and 11.

User #1 went ahead and ran the system without any problem by double-clicking on the icon. The program executed as expected. The user examined the program interface for a couple of minutes before starting any sort of interaction. The user seemed to understand how the system worked at first glance, so read task 1 and began the interaction.

The user faced a slight difficulty when constructing the first matrix but this only happened with the first element. After filling in the first element the user constructed successfully both matrices at a good pace. Having constructed the matrices the user went ahead and performed the addition by clicking on the "A + B" button.

Task 1: completed.

The user continued by examining task 4 and immediately tested his whit, by trying to re-use the same matrices as for task 1. So went straight to the multiplication button and obtained a result. The user expressed clear signs of satisfaction not having to re-construct the matrices.

Task 4: completed.

The user successfully managed to alter the size of the matrices from 4 x 4 to 5 x 5 and went ahead to construct both matrices. He performed the addition once again without any difficulty. When it came to the final part of the task, calculating the determinant of the result, the user hesitated and seemed puzzled. He referred to the user guide available to him and resumed the interaction to complete the task successfully following the instructions of the user guide.

Task 11: completed.

What is your overall opinion after having used the system? Any suggestions?

I think that it was fairly straight forward to complete the tasks that I was required to carry out. I had some difficulty with one of the tasks but I used the user guide that was very approachable and helped me thorough the task.
I found the system very user-friendly as there weren't any obscure features. If anything, having to delete the matrix element before entering the ones I wanted at the beginning was a bit of a hassle, but that's just me being a bit lazy I must say. Also, maybe the user guide could be included as a help menu.

User #2

Do you have a mathematical background? Have you performed matrix calculations before?

I did GCSE maths, that's about it. Yes, some basic ones.

Have you used any computer program to solve matrix calculations?

No.

User #1 was requested to complete tasks 2, 3 and 6.

The user ran the system without any trouble. The user had to be directed in order for him to be able input the matrix elements. Once the first element was introduced he had no trouble filling in the rest. He clicked on the operation button and performed the calculation. An error popped-up, the matrices were of different size. The user clicked ok and went to change the inputs. Performed the operation again and obtained a result.

Task 2: Completed.

The user followed on to task 3. He used the same matrices so just clicked on the multiplication operation. (This was used as a valid step). The result was displayed.

Task 3: Completed.

The user changed the matrix size correctly to a 5 x 5 matrix and followed on to filling it up. Clicked on the determinant button and obtained the result.

Task 6: Completed.

What is your overall opinion after having used the system? Any suggestions?

I found the system fairly easy to use. It would be good if it had a help system to access as you are performing an operation. I think this program is quite useful for getting the results quickly.

User #3

Do you have a mathematical background? Have you performed matrix calculations before?

Yes, I'm doing a degree in maths.

Have you used any computer program to solve matrix calculations?

Yes, a few.

User #1 was requested to complete tasks 10, 12 and 13.

The user ran the system without any trouble and made her way through task 10. The user entered the matrices and performed the subtraction without any trouble. The user got a bit confused about the next step, and made use of the user guide. Having read the instructions followed on to finish the task.

Task 10: Completed.

The user entered matrix A to be a 2 x 3 matrix and matrix B to be a 3 x 1 matrix then clicked on the multiplication button and obtained the result. Then she carried on to swap the matrices which was done successfully and performed the multiplication again. As expected an error occurred, in this case the multiplication was not possible.

Task 12: completed.

The user entered the matrix correctly and performed the inverse by clicking on the inverse button. She successfully made matrix B equal the result and then just performed the multiplication. The identity matrix was the result.

Task 13: completed.

What is your overall opinion after having used the system? Any suggestions?

It's different form any other program I've used before. It is straight to the point. It would be good if more operations could be performed. E.g. the rank.

A5 User Guide

User Guide

This user guide will help you understand and work through j-LAS so that you can perform your matrix operations.

Table of contents

1. Running the program. 86

2. Performing matrix operations. 86

2.1. Initial state. 86

2.2. Constructing matrices. 86

2.3. Swapping matrices. 87

2.4. Adding matrices. 87

2.5. Subtracting matrix B from matrix A. 88

2.6. Multiplying matrices using naïve algorithm. 88

2.7. Multiplying matrices using Strassen's algorithm (for square matrices only). 88

2.8. Finding the inverse of a matrix. 88

2.9. Finding determinant of matrices 89

2.10. Solving systems of linear equations. 89

2.11. Using the result matrix. 89

3. Finalizing the application. 90

1. Running the program.

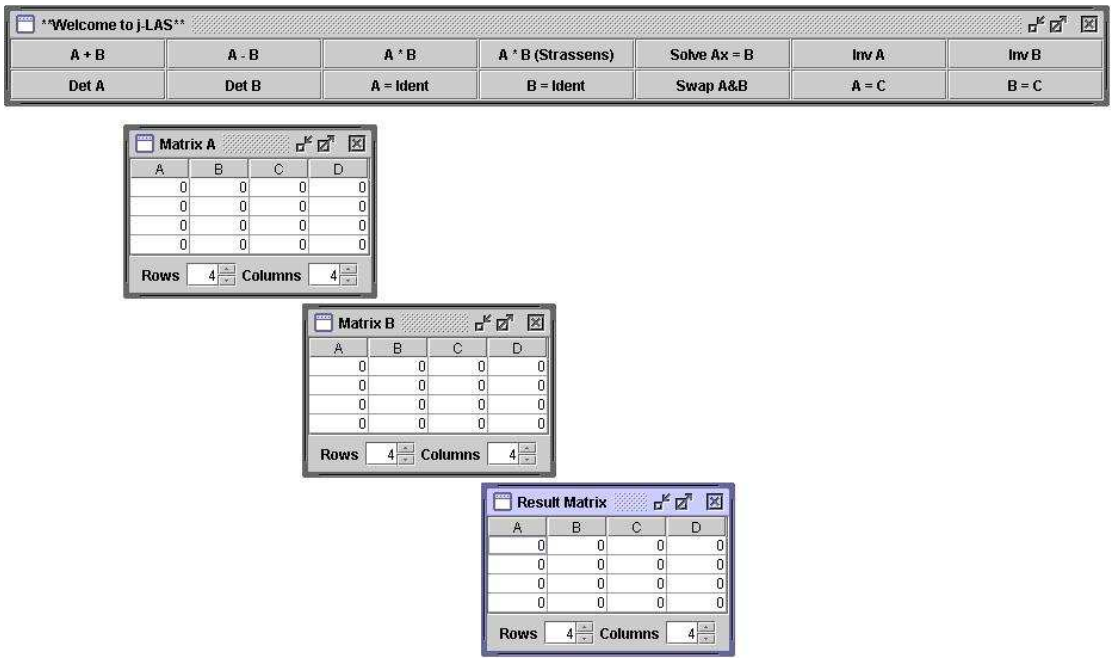
1. Insert the CD in your CD-ROM drive (Typically, D:\).

2. Open the folder “Program”

3. Double click on “j-LAS”

2. Performing matrix operations.

2.1. Initial state.



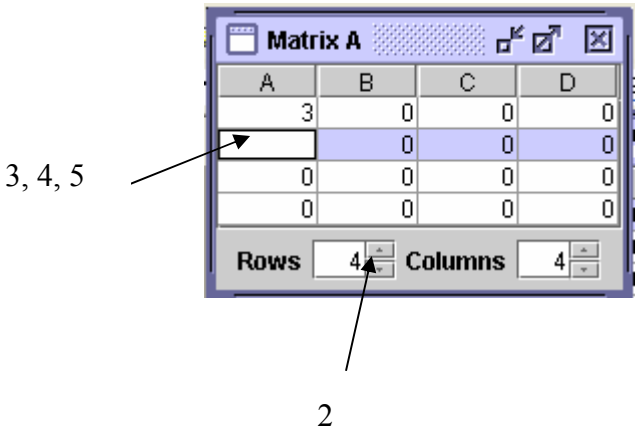
When you run j-LAS, this is the initial state of the program. All the windows can be expanded and stretched; so if you have large matrices you can expand or stretch the matrix windows to have an optimum view of the matrix you are dealing with.

2.2. Constructing matrices.

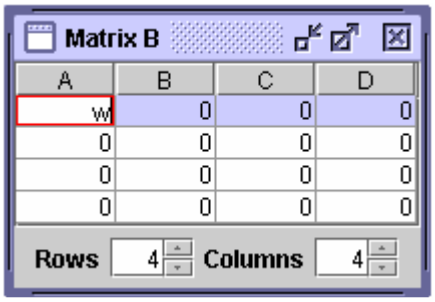
To construct matrices:

1. Decide which matrix is being constructed, A or B.

2. Select the number of rows and columns of the matrix by clicking on the up and down arrows at the bottom of the matrix window.
3. Double-click on the appropriate element box.
4. Delete the existing content, which will be 0 in the initial matrix, and input the appropriate value.
5. Press enter or double-click on next element to validate entry.
6. Repeat procedure 5 to complete the matrix.
7. Once all elements the have been validated the matrix is ready to perform operations.



Note: All the values inputted into the matrices must be numbers. The matrices accept values up to two decimal places. If you were to input anything different from a number the element box will highlight in red and not let you validate it until a number is entered.



2.3. Swapping matrices.

If you desire to swap existing matrices A and B, just click on the

button in the control panel and the matrix contents will swap.

2.4. Adding matrices.

1. Construct matrices A and B appropriately.
2. Click on

button in the control panel.
3. The result will be displayed in the “Result Matrix”

Note: If the sizes of the matrices does not coincide an error message will pop-up. Click OK, and amend the matrices correspondingly.

2.5. Subtracting matrix B from matrix A.

1. Construct matrices A and B appropriately.
2. Click on

button in the control panel.
3. The result will be displayed in the "Result Matrix".

Note: If the sizes of the matrices does not coincide an error message will pop-up. Click OK, and amend the matrices correspondingly.

2.6. Multiplying matrices using naïve algorithm.

1. Construct matrices A and B appropriately.
2. Click on

button in the control panel.
3. The result will be displayed in the "Result Matrix".

Note: If the sizes of the matrices are not appropriate to perform multiplication (i.e. # rows of matrix A = # cols. Of matrix B) an error message will pop-up. Click OK, and amend the matrices correspondingly.

2.7. Multiplying matrices using Strassen's algorithm (for square matrices only).

1. Construct matrices A and B appropriately.
2. Click on

button in the control panel.
3. The result will be displayed in the "Result Matrix".

Note: If the sizes of the matrices are not appropriate to perform multiplication (i.e. # rows of matrix A = # cols. Of matrix B) an error message will pop-up. Click OK, and amend the matrices correspondingly.

2.8. Finding the inverse of a matrix.

2.8.1. Inverse of matrix A.

1. Construct matrix A.
2. Click on

button in the control panel.
3. The result will be displayed in the "Result Matrix".

2.8.2. Inverse of matrix B.

1. Construct matrix B.

2. Click on Inv B button in the control panel.
3. The result will be displayed in the “Result Matrix”.

2.9. Finding determinant of matrices

2.9.1. Determinant of matrix A.

1. Construct matrix A.
2. Click on Det A button in the control panel.
3. The result will be displayed in the “Result Matrix”.

2.9.2. Determinant of matrix B.

1. Construct matrix B.
2. Click on Det B button in the control panel.
3. The result will be displayed in the “Result Matrix”.

2.10. Solving systems of linear equations.

It is possible to solve system of equations of the sort Ax = B. Let’s look at an example of a system of linear equations.

$$\begin{aligned} x_1 + x_2 + x_3 &= 2 \\ 2x_1 + 3x_2 + x_3 &= 3 \\ x_1 - x_2 - 2x_3 &= -6 \end{aligned}$$

this system of linear equations translates to the form Ax = B as follows:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 1 & -1 & -2 \end{pmatrix} x = \begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix}$$

So, in order to solve a system of linear equations:

1. Construct matrices A and B according to the system of linear equations.
2. Click on the Solve Ax = B button in the control panel.
3. The result will be displayed in the “Result Matrix”.

Note: Matrix A must be a square matrix. If it isn’t a square matrix an error message will pop-up. Click OK, and amend the matrix correspondingly. *Don’t forget matrix B exception*.

2.11. Using the result matrix.

You may need to use the result matrix to continue performing matrix operations. It is possible to copy the result matrix into Matrix A or B without having to construct the matrix again manually. In order to copy the result matrix into matrix A click on the

button in the control panel and to copy the result matrix into matrix B click on the

button.

3. Finalizing the application.

In order to exit the application, click on cross at the top right-hand corner of any of the windows or press Alt + F4 on the keyboard and the application will shut down.